Programmazione IEC 61131-3 (LogicLab)

ELSIST S.r.l.
Sistemi in elettronica

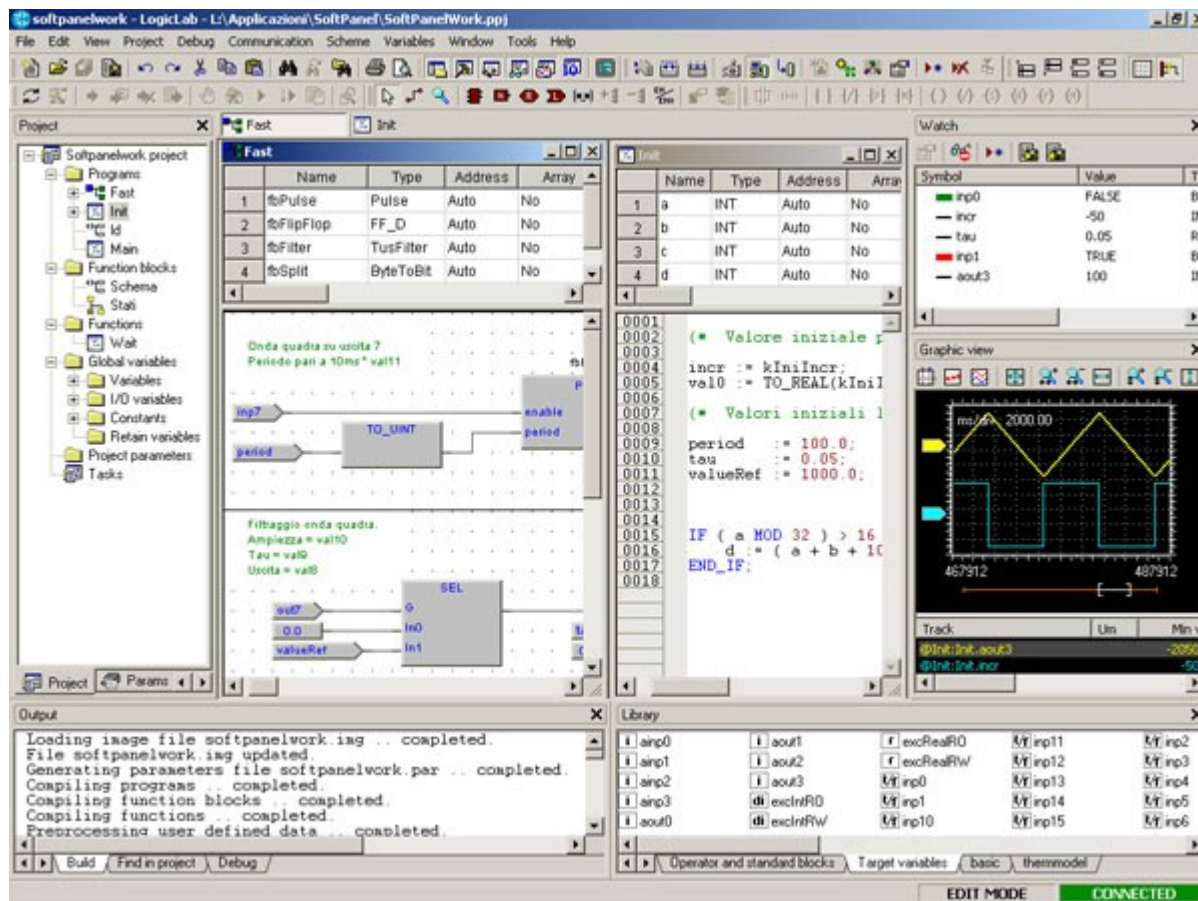
Via G. Brodolini, 15 (Z.I.)
15033 CASALE M.TO
ITALY

Internet: http://www.elsist.it
Email: elsist@elsist.it

TEL. (39)-0142-451987
FAX (39)-0142-451988

INDICE

1 LogicLab

LogicLab è un compilatore PLC IEC61131-3 che supporta tutti i 5 linguaggi dello standard, la facilità e l'immediatezza d'uso degli editor grafici e testuali, le funzioni di drag & drop estese a tutti i contesti del framework, le diverse utility integrate ed i debugger grafici e testuali rendono LogicLab un ambiente di sviluppo efficiente e particolarmente gradevole da utilizzare.

Il compilatore di LogicLab genera direttamente il codice macchina per il processore del sistema target, garantendo una alta efficienza prestazionale.Il tool è stato sviluppato da Axel, una azienda italiana con pluriennale esperienza nella produzione di software per automazione industriale, e personalizzato per l'utilizzo con i nostri dispositivi.

2 Risorse del sistema

Gli I/O logici sono automaticamente gestiti in immagine di processo dal sistema operativo che provvede a trasferire lo stato di tutti gli ingressi logici nella immagine degli ingressi in memoria di sistema ed a trasferire il valore presente nella immagine delle uscite dalla memoria di sistema alle uscite logiche.

Quindi testando lo stato della immagine di memoria degli ingressi logici si testa lo stato del relativo punto di ingresso (Esempio ***IX0.0*** corrisponde all'ingresso 0 del modulo 0, ***IX1.5*** corrisponde all'ingresso 5 del modulo 1).

Scrivendo lo stato nella immagine di memoria delle uscite logiche si setta lo stato del relativo punto di uscita (Esempio ***QX0.0*** corrisponde all'uscita 0 del modulo 0, ***QX1.5*** corrisponde all'uscita 5 del modulo 1).

Gli ***I/O logici*** possono anche essere gestiti tramite le funzioni ***SysGetPhrDI*** e ***SysGetPhrDO***.

Gli ***I/O analogici*** sono gestiti tramite le funzioni ***SysGetAnInp*** e ***SysSetAnOut***

I ***contatori*** sono acquisiti dal blocco funzione ***SysGetCounter***.

Gli ingressi ***encoder*** sono acquisiti dal blocco funzione ***SysGetEncoder***.

Il ***CAN bus*** viene gestito dalle funzioni ***SysCANRxMsg*** e ***SysCANTxMsg***.

Per accedere alle ***porte seriali*** occorre utilizzare la funzione ***Sysfopen*** definendo il nome della porta da utilizzare. Esistono moduli di estensione che sono provvisti di porte seriali, l'accesso a queste porte è esattamente uguale a quello delle porte presenti sul modulo CPU. Si utilizza la definizione ***PCOMx.y*** dove con ***x*** si indica l'indirizzo del modulo e con ***y*** il numero di porta presente sul modulo. (Esempio ***PCOM0.0*** definisce la porta 0 presente sul modulo 0, ***PCOM1.2*** definisce la porta 2 presente sul modulo 1 e così via).

Per accedere al file system (Sia sul disco interno al sistema che su SD Card) occorre utilizzare la funzione ***Sysfopen***.

Nei modelli provvisti di porta Ethernet è possibile gestire connessioni sia UDP che TCP/IP utilizzando la funzione ***Sysfopen***.

2.1 Architettura memoria

La memoria del sistema è così suddivisa:

DB	Dimensione	Descrizione
IX0	32 Bytes	Ingressi logici modulo 00 (R)
IX1	32 Bytes	Ingressi logici modulo 01 (R)
IX2	32 Bytes	Ingressi logici modulo 02 (R)
IX3	32 Bytes	Ingressi logici modulo 03 (R)
IX4	32 Bytes	Ingressi logici modulo 04 (R)
IX5	32 Bytes	Ingressi logici modulo 05 (R)
IX6	32 Bytes	Ingressi logici modulo 06 (R)
IX7	32 Bytes	Ingressi logici modulo 07 (R)
IX8	32 Bytes	Ingressi logici modulo 08 (R)
IX9	32 Bytes	Ingressi logici modulo 09 (R)
IX10	32 Bytes	Ingressi logici modulo 10 (R)
IX11	32 Bytes	Ingressi logici modulo 11 (R)
IX12	32 Bytes	Ingressi logici modulo 12 (R)
IX13	32 Bytes	Ingressi logici modulo 13 (R)
IX14	32 Bytes	Ingressi logici modulo 14 (R)
IX15	32 Bytes	Ingressi logici modulo 15 (R)
IX255	32 Bytes	Ingressi logici modulo CPU (R)
QX0	32 Bytes	Uscite logiche modulo 00 (R/W)
QX1	32 Bytes	Uscite logiche modulo 01 (R/W)
QX2	32 Bytes	Uscite logiche modulo 02 (R/W)
QX3	32 Bytes	Uscite logiche modulo 03 (R/W)
QX4	32 Bytes	Uscite logiche modulo 04 (R/W)
QX5	32 Bytes	Uscite logiche modulo 05 (R/W)
QX6	32 Bytes	Uscite logiche modulo 06 (R/W)
QX7	32 Bytes	Uscite logiche modulo 07 (R/W)
QX8	32 Bytes	Uscite logiche modulo 08 (R/W)
QX9	32 Bytes	Uscite logiche modulo 09 (R/W)
QX10	32 Bytes	Uscite logiche modulo 10 (R/W)
QX11	32 Bytes	Uscite logiche modulo 11 (R/W)
QX12	32 Bytes	Uscite logiche modulo 12 (R/W)
QX13	32 Bytes	Uscite logiche modulo 13 (R/W)
QX14	32 Bytes	Uscite logiche modulo 14 (R/W)
QX15	32 Bytes	Uscite logiche modulo 15 (R/W)
QX255	32 Bytes	Uscite logiche modulo CPU (R/W)
MX0	512 Bytes	Variabili di sistema sola lettura (R)
MX1	512 Bytes	Variabili di sistema lettura/scrittura (R/W)
MX100	4096 Bytes	Memoria utente (R/W). Da indirizzo 2048 a 4095 i dati sono ritentivi.

2.2 Memoria di backup (Retain)

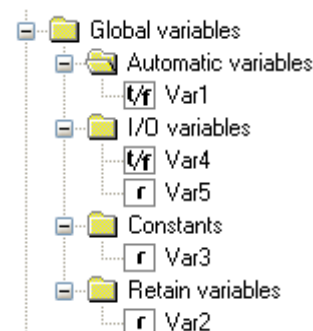
SlimLine dispone di 2048 bytes di memoria ritentiva nell'area memoria utente ***MX100*** ed ulteriori 2000 bytes di memoria ritentiva a disposizione utente per allocare variabili mnemoniche.

Le variabili allocate nella memoria utente ***MX100*** da indirizzo 2048 ad indirizzo 4095 sono ritentive, mantengono il loro valore anche allo spegnimento del sistema.

Qualsiasi variabile mnemonica a cui viene attribuito l'attributo ***RETAIN***, manterrà il suo valore anche allo spegnimento del sistema. Da quanto detto precedentemente l'area totale allocabile per le variabili ***RETAIN*** è di 2000 bytes.

	Name	Type	Address	Group	Array	Init value	Attribute	Description
1	Var1	BOOL	Auto		No	FALSE	..	Sample variable 1
2	Var2	REAL	Auto		No	0	RETAIN	Sample variable 2
3	Var3	REAL	Auto		No	12.5	CONSTANT	Sample variable 3
4	Var4	BOOL	%MD100.0		No	FALSE	..	Sample variable 4
5	Var5	REAL	%MD100.2048		No	0	..	Sample variable 5

Come si vede dalla foto la variabile ***Var2*** è dichiarata con l'attributo ***RETAIN*** e manterrà il suo valore anche allo spegnimento del sistema. La variabile ***Var5*** allocata nella memoria utente ***MD100.2048*** pur essendo ritentiva non necessita dell'attributo ***RETAIN*** in quanto è implicito dalla sua allocazione.

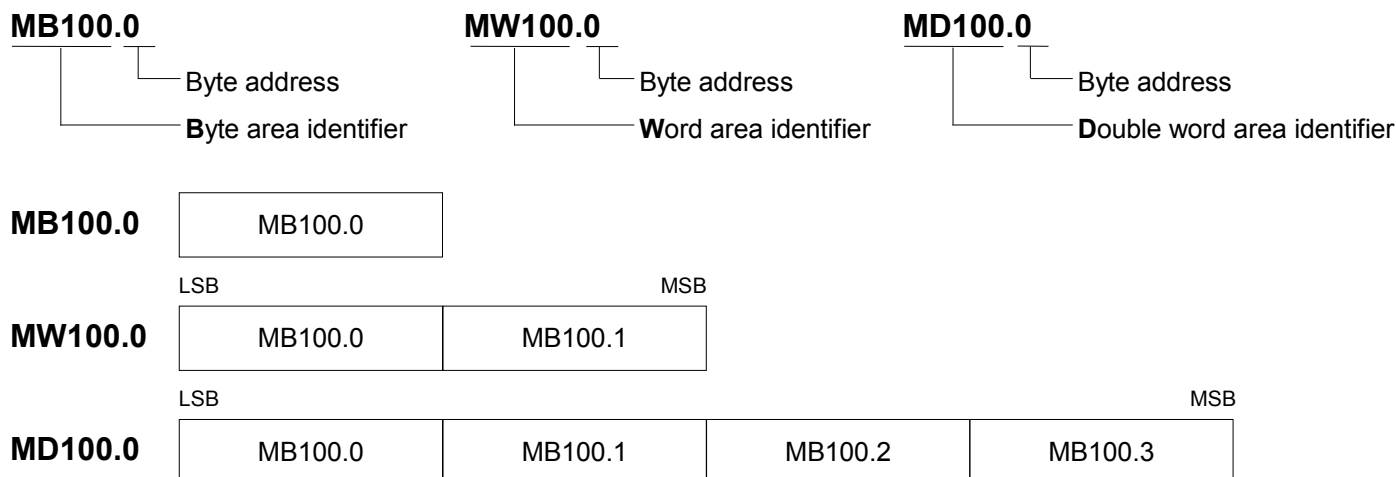
Nella finestra di navigazione progetto, tutte le variabili globali sono suddivise in base alla loro definizione, e come si nota nella cartella delle variabili ritentive figureranno solo le variabili mnemoniche ***Var2*** e non le variabili allocate nella memoria utente ***Var5*** pur essendo anch'essa di tipo ritentivo.

2.3 Accesso alla memoria

IX: Immagine di processo ingressi logici
SlimLine esegue la lettura degli ingressi logici all'inizio di ogni loop di esecuzione programma. E' possibile accedere a quest'area utilizzando variabili di tipo ***BOOL***, ogni indirizzo rappresenta lo stato booleano del relativo ingresso logico. L'indirizzo ***IX0.0***, rappresenta lo stato dell'ingresso 0 del modulo 0, l'indirizzo ***IX5.12***, rappresenta lo stato dell'ingresso 12 del modulo 5.

QX: Immagine di processo uscite logiche
SlimLine esegue la scrittura delle uscite logiche alla fine di ogni loop di esecuzione programma. E' possibile accedere a quest'area utilizzando variabili di tipo ***BOOL***, ogni indirizzo rappresenta lo stato booleano della relativa uscita logica. L'indirizzo ***QX0.0***, rappresenta lo stato dell'uscita 0 del modulo 0,l'indirizzo ***QX5.12***, rappresenta lo stato dell'uscita 12 del modulo 5.

MX: Area di memoria
A queste aree è possibile accedere utilizzando tutti i tipi di variabili definiti. Siccome tutte le variabili utilizzano la stessa area di memoria, occorre prestare attenzione alla dimensione in bytes del tipo definito per evitare sovrapposizioni di indirizzo.

Ad esempio una variabile ***DWORD*** allocata ad indirizzo ***MX100.10*** utilizzerà anche lo spazio di memoria ***MX100.11***, ***MX100.12*** ed ***MX100.13***. Quindi allocando una variabile ***BYTE*** all'indirizzo ***MX100.11*** si andrebbe ad occupare uno spazio di memoria già utilizzato dalla variabile precedente.

E' comunque possibile allocare variabili sovrapponendone l'indirizzo, esempio allocare due variabili ***BYTE*** sugli stessi indirizzi di una variabile ***WORD*** per andarne a considerare la parte MSB od LSB. Oppure allocare due variabili ***WORD*** sugli stessi indirizzi di una variabile ***DWORD*** per andarne a considerare la parte MSW od LSW. Riporto una semplice tabella esplicativa.

Attenzione! SlimLine è basato su architettura ARM e questo tipo di architettura assume che:

Le variabili a 16 bits, ***WORD***, ***INT***, ***UINT*** siano allocate in memoria ad indirizzi ***divisibili per 2***. Quindi una variabile a 16 bits potrà assere allocata ad esempio ad indirizzo MW100.32 ma non ad indirizzo MW100.33.

Le variabili a 32 bits ***DWORD***, ***DINT***, ***UDINT***, ***REAL*** siano allocate ad indirizzi ***divisibili per 4***. Quindi una variabile a 32 bits potrà assere allocata ad esempio ad indirizzo MD100.32 ma non ad indirizzo MD100.33, MD100.34, MD100.35.

Questa regola viene applicata automaticamente dal compilatore di LogicLab anche per quanto riguarda le ***strutture dati***, quindi nella definizione di una struttura dati composta da variabili di tipo eterogeneo LogicLab inserirà automaticamente dei bytes di spaziatura per allineare correttamente le variabili di a 16 bits ed a 32 bits.

3 Definizione tipo dati

Oltre ai dati standard definiti dalla normativa IEC61131 sono stati definiti altri tipi di dato che possono essere utilizzati nel programma PLC.

3.1 FILEP, file pointer

Questo tipo di dati è utilizzato dalle funzioni che eseguono accesso alle risorse di I/O del sistema, una variabile di tipo ***FILEP*** punta ad una risorsa utilizzata per effettuare la lettura e/o scrittura di dati. Un esempio di file pointer è il puntatore ad una porta seriale od un file su disco.

3.2 SYSSERIALMODE, modo comunicazione porta seriale

Questo tipo di dati è utilizzato dalle funzioni che eseguono lettura ed impostazione modo di comunicazione su porta seriale. Il tipo dati contiene tutte le informazioni per caratterizzare la comunicazione sulla porta seriale.

Name	Type	Description
Baudrate	UDINT	Baud rate porta seriale (300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200).
Parity	STRING[1]	Tipo di parità, valori possibili “E” pari, “O” dispari, “N” nessuna.
DataBits	USINT	Numero di bit frame dato, valori possibili 7, 8.
StopBits	USINT	Numero di bit di stop, valori possibili 1, 2.
DTRManagement	USINT	Modo di gestione del segnale DTR sulla porta seriale, vedi Serial mode definition.
DTRComplement	BOOL	FALSE: DTR normale, TRUE: DTR complementato.
EchoFlush	BOOL	FALSE: I dati trasmessi sono ritornati in ricezione. TRUE:I dati trasmessi sono ignorati, su comunicazioni in RS485.
DTROnTime	UINT	Tempo di attesa trasmissione caratteri su porta seriale dopo attivazione segnale DTR (mS). Questo parametro assume significato solo se ***DTRManagement*** è impostato nel modo ***DTR_AUTO_W_TIMES***, vedi Serial mode definition.
DTROffTime	UINT	Tempo di attesa dopo trasmissione ultimo dato e disattivazione segnale DTR (mS). Questo parametro assume significato solo se ***DTRManagement*** è impostato nel modo ***DTR_AUTO_W_TIMES***, vedi Serial mode definition.

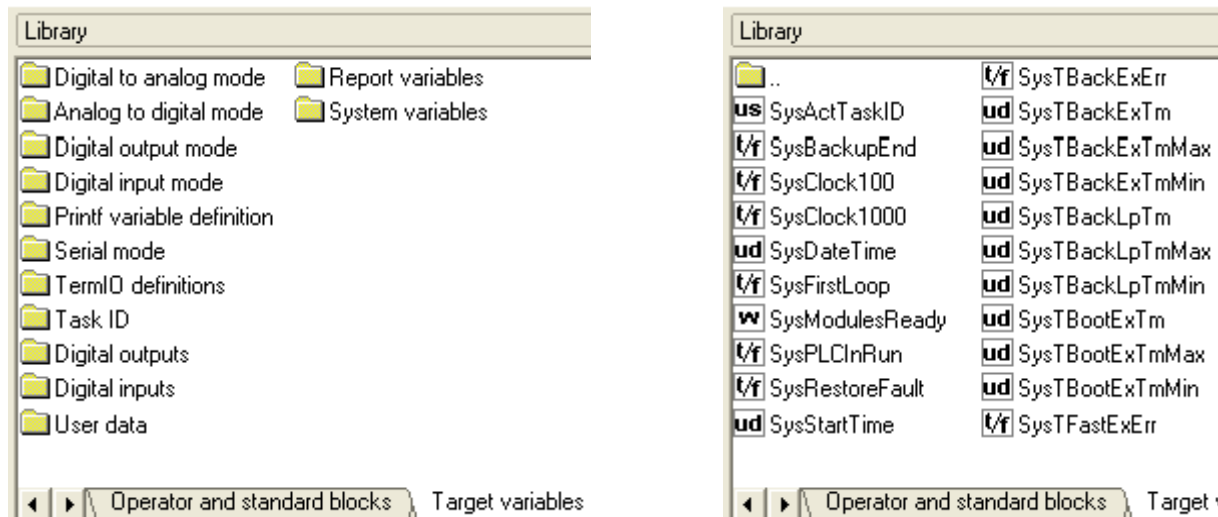
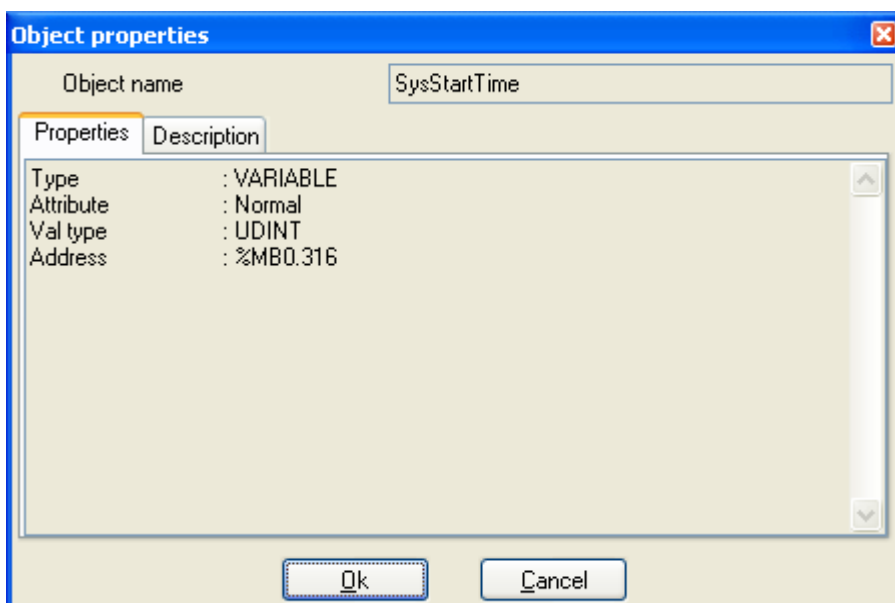
3.3 SYSCANMESSAGE, messaggio CAN

Questo tipo di dati è utilizzato dalle funzioni che gestiscono il controller CAN. La struttura definisce il formato di un messaggio CAN.

Name	Type	Description
RmReq	BOOL	FALSE:Data frame, TRUE:Remote request.
Length	USINT	Lunghezza record dati da 0 a 8 bytes.
MsgID	UDINT	Message ID, 11 o 29 bit di identificativo messaggio. Il bit 31 è il bit di FF.
Data	ARRAY[0..7] OF USINT	Array dati messaggio

4 Variabili di sistema

Con l'ambiente di sviluppo LogicLab vengono pubblicate variabili di sistema che possono essere referenziate nel programma per accedere ad informazioni sul sistema. Le variabili sono visualizzate da LogicLab nella finestra delle librerie.

Se la finestra non è visualizzata, occorre abilitarne la visualizzazione dalla voce di menù ***Wiew*** → ***Tool windows*** → ***Library***. Attivando il Tab ***Target variables*** verrà visualizzato un elenco con tutte le variabili pubblicate suddivise in cartelle. Un doppio click del mouse sulla cartella ***System variables*** si aprirà la cartella visualizzando tutte le variabili pubblicate (Vedi foto a destra).

Agendo con il tasto destro del mouse su ogni singola variabile è possibile visualizzarne la finestra delle proprietà in cui è indicato il tipo e l'indirizzo di allocazione, cosi come nella figura a lato.

4.1 Variabili sola lettura (System variables)

Queste variabili di sola lettura presenti nella cartella ***System variables***, ritornano informazioni sul sistema. Il programma utente può utilizzare le variabili a piacere ma non può modificarne il valore.

Name	Type	Description
SysClock100	BOOL	Clock lampeggiante con duty cycle di 100 mS.
SysClock1000	BOOL	Clock lampeggiante con duty cycle di 1 S.
SysBackupEnd	BOOL	Attiva per un loop di programma al termine di un ciclo di backup dati.
SysRestoreFault	BOOL	Si attiva alla accensione se i dati di backup sono in errore. Tutti i dati sono azzerati.
SysPLCInRun	BOOL	Sempre attiva.
SysFirstLoop	BOOL	Attiva per un loop alla prima esecuzione di ogni task PLC.
SysLLabCn	BOOL	Attiva se l'ambiente di sviluppo LogicLab è connesso al sistema.
SysIsDST	BOOL	Attiva se si è nel periodo di Daylight Saving Time
SysDTSet	BOOL	Attiva per un loop esecuzione task background su modifica real time clock da sistema operativo.
SysUVSet	BOOL	Attiva per un loop esecuzione task background su modifica impostazioni utente da sistema operativo, vedi esempio.
SysAlwaysOff	BOOL	Variabile mai attiva
SysAlwaysOn	BOOL	Variabile sempre attiva
SysFFBUspt	BOOL	Il programma esegue almeno una funzione o blocco funzione non supportato
SysFFBPrt	BOOL	Il programma esegue almeno una funzione o blocco funzione protetto
SysActTaskID	USINT	Numero di identificazione della task in corso, vedi tipi definiti.
SysErActTaskID	USINT	Numero di identificazione della task in cui l'errore si è verificato, vedi tipi definiti.
SysModulesReady	UINT	Ogni bit della variabile se attivo indica la presenza del modulo connesso al bus SlimLine.
SysApllVMajor	UINT	Valore Major della versione dell'applicazione.
SysApllVMinor	UINT	Valore Minor della versione dell'applicazione.
SysTBackLpTm	UDINT	Tempo di loop attuale della task di background PLC (uS).
SysTBackLpTmMin	UDINT	Tempo di loop minimo della task di background PLC (uS). E' possibile inizializzarne il valore settando il bit SysTimeInit.
SysTBackLpTmMax	UDINT	Tempo di loop massimo della task di background PLC (uS). E' possibile inizializzarne il valore settando il bit SysTimeInit.
SysTBootExTm	UDINT	Tempo di esecuzione attuale della task di boot PLC (uS).
SysTBootExTmMin	UDINT	Tempo di esecuzione minimo della task di boot PLC (uS). E' possibile inizializzarne il valore settando il bit SysTimeInit.
SysTBootExTmMax	UDINT	Tempo di esecuzione massimo della task di boot PLC (uS). E' possibile inizializzarne il valore settando il bit SysTimeInit.
SysTFastExTm	UDINT	Tempo di esecuzione attuale della task fast PLC (uS).
SysTFastExTmMin	UDINT	Tempo di esecuzione minimo della task fast PLC (uS). E' possibile inizializzarne il valore settando il bit SysTimeInit.
SysTFastExTmMax	UDINT	Tempo di esecuzione massimo della task fast PLC (uS). E' possibile inizializzarne il valore settando il bit SysTimeInit.
SysTSlowExTm	UDINT	Tempo di esecuzione attuale della task slow (uS).
SysTSlowExTmMin	UDINT	Tempo di esecuzione minimo della task slow (uS). E' possibile inizializzarne il valore settando il bit SysTimeInit.
SysTSlowExTmMax	UDINT	Tempo di esecuzione massimo della task slow (uS). E' possibile inizializzarne il valore settando il bit SysTimeInit.
SysTBackExTm	UDINT	Tempo di esecuzione attuale della task background (uS).
SysTBackExTmMin	UDINT	Tempo di esecuzione minimo della task background (uS). E' possibile inizializzarne il valore settando il bit SysTimeInit.
SysTBackExTmMax	UDINT	Tempo di esecuzione massimo della task background (uS). E' possibile inizializzarne il valore settando il bit SysTimeInit.
SysStartTime	UDINT	Data ed ora di start esecuzione programma PLC (Epoch time).
SysTime	UDINT	Tempo di sistema si incrementa ad ogni 1 mS raggiunto il valore massimo si reinizializza.

Name	Type	Description
SysTFastLpTm	UDINT	Tempo di loop della task fast PLC. Il tempo è impostabile tramite la funzione SysSetTaskLpTime.
SysTSlowLpTm	UDINT	Tempo di loop della task slow PLC. Il tempo è impostabile tramite la funzione SysSetTaskLpTime.
SysApplID	UDINT	ID applicazione, è un numero univoco che identifica il programma utente attualmente in esecuzione sul sistema.
SysMfcCode	UDINT	Codice manufacturer, questo codice và richiesto con il prodotto. Se non definito è ritornato il codice ***0***.
SysCustomerCode	UDINT	Codice customer, questo codice può essere impostato dall'utente che ha accesso come amministratore al sistema. Se non definito è ritornato il codice ***0***.
SysErCode	UDINT	Numero dell'errore di esecuzione programma.
SysErTime	UDINT	Data ed ora in cui l'errore è avvenuto (Epoch time).
SysSerialNr	UDINT	Numero di matricola del prodotto.
SysApllBTime	UDINT	Build time (Epoch time) della dell'applicazione.
SysUniqueID	UDINT	ID univoco prodotto, vedi note.
SysCode	STRING[10]	Codice del prodotto.
SysFVersion	STRING[10]	Versione firmware del prodotto.
SysErInfos	STRING[32]	Informazioni aggiuntive sull'errore.
SysApllName	STRING[10]	Nome dell'applicazione.
SysUSetA	STRING[16]	Valore impostato da utente in variabile Set(A), vedi esempio.
SysUSetB	STRING[16]	Valore impostato da utente in variabile Set(B), vedi esempio.
SysUSetC	STRING[16]	Valore impostato da utente in variabile Set(C), vedi esempio.
SysUSetD	STRING[16]	Valore impostato da utente in variabile Set(D), vedi esempio.

4.2 Variabili lettura e scrittura (System variables)

Queste variabili lettura e scrittura presenti nella cartella ***System variables***, permettono di leggere e scrivere da programma utente informazioni sul funzionamento del sistema. Il programma utente può utilizzare le variabili a piacere ed anche modificarne il valore.

Name	Type	Description
SysTimeInit	BOOL	Attivata da programma utente o da debug permette di inizializzare il calcolo dei tempi di loop e di esecuzione delle task PLC. La variabile viene automaticamente resettata dal sistema.
SysDateTime	UDINT	Data ed ora di sistema (Epoch time). Modificando il valore verrà automaticamente aggiornato anche il real time clock.
SysLastError	UDINT	Last error, ritorna il valore dell'ultimo errore verificatosi nella esecuzione di una funzione o di un blocco funzione, tabella ID oggetti.
SysUInfoA	STRING[16]	Valore ritornato ad utente in variabile Info(A), vedi esempio.
SysUInfoB	STRING[16]	Valore ritornato ad utente in variabile Info(B), vedi esempio.
SysUInfoC	STRING[16]	Valore ritornato ad utente in variabile Info(C), vedi esempio.
SysUInfoD	STRING[16]	Valore ritornato ad utente in variabile Info(D), vedi esempio.

4.3 ID univoco prodotto

Ogni prodotto ha un ID univoco che è ritornato nella variabile ***UniqueID***, il valore è ottenuto in base al tipo di prodotto ed al suo numero di matricola. La formula per il calcolo dell'UniqueID è la seguente:

UniqueID=(131072*PType)+Serial number

Ad esempio un MPS050A030 con matricola 125 avrà UniqueID=1310845. Ecco la tabella ***Ptype***.

PType	*Code*	*Description*
0000	MPS046A000	SlimLine (Lite version)
0001	MPS046A100	SlimLine (Rs485 version)
0002	MPS046A200	SlimLine (CAN version)
0003	MPS048A100	SlimLine ARM9 RS485 (Linux Open)
0004	MPS048A200	SlimLine ARM9 CAN (Linux Open)
0005	MPS049A100	SlimLine ARM9 RS485 (Linux PLC)
0006	MPS049A200	SlimLine ARM9 CAN (Linux PLC)
0007	MPS050A000	SlimLine Low Cost ARM7 (Vers. Lite)
0008	MPS050A010	SlimLine Low Cost ARM7 (Vers. Base)
0009	MPS050A020	SlimLine Low Cost ARM7 (Vers. Full RS485)
0010	MPS050A030	SlimLine Low Cost ARM7 (Vers. Full CAN)
0011	PCB123B000	SlimLine OEM (Lite version)
0012	PCB123B100	SlimLine OEM (Rs485 version)
0013	PCB123B200	SlimLine OEM (CAN version)
0014	MPS046B000	SlimLine (Lite version)
0015	MPS046B100	SlimLine (Rs485 version)
0016	MPS046B200	SlimLine (CAN version)
0017	PCB123D000	SlimLine OEM (Lite version)
0018	PCB123D100	SlimLine OEM (Rs485 version)
0019	PCB123D200	SlimLine OEM (CAN version)
0020	PCB131A000	SlimLine ARM7 Compact Relay CPU Board (Lite vers.)
0021	PCB131A010	SlimLine ARM7 Compact Relay CPU Board (Base vers.)
0022	PCB131A020	SlimLine ARM7 Compact Relay CPU Board (Full RS485 v.)
0023	PCB131A030	SlimLine ARM7 Compact Relay CPU Board (Full CAN v.)
0024	MPS051A000	Netlog III Base Relay
0025	MPS051A001	Netlog III Full Relay (Rs485 version)
0026	MPS051A011	Netlog III Full Relay and Display (Rs485 version)
0027	MPS051A002	Netlog III Full Relay (CAN version)
0028	MPS051A012	Netlog III Full Relay and Display (CAN version)
0029	MPS051A300	Netlog III Base Static
0030	MPS051A301	Netlog III Full Static (Rs485 version)
0031	MPS051A311	Netlog III Full Static and Display (Rs485 version)
0032	MPS051A302	Netlog III Full Static (CAN version)
0033	MPS051A312	Netlog III Full Static and Display (CAN version)

5 Definizioni dati

Oltre alle cartelle delle variabili di sistema sono presenti anche cartelle con identificatori di tipo dati che permettono di identificare in modo unifico un dato di sistema.

5.1 Variable types definition, definizione tipo variabili

Ogni tipo di variabile è definito con un valore che lo identifica, il valore è indicato con definizioni che si possono trovare nella cartella ***Variable types definition***.

Define	Type	Value	Description
BOOL_TYPE	USINT	10	Variabile booleana (BOOL), 1 bit può assumere solo significato FALSE o TRUE.
BYTE_TYPE	USINT	20	Variabile byte (BYTE) 8 bits senza segno, range da 0 a 255.
SINT_TYPE	USINT	21	Variabile byte (SINT) 8 bits con segno, range da -128 a +127.
USINT_TYPE	USINT	22	Variabile byte (USINT) 8 bits senza segno, range da 0 a 255.
WORD_TYPE	USINT	30	Variabile word (WORD) 16 bits senza segno, range da 0 a 65535.
INT_TYPE	USINT	31	Variabile word (INT) 16 bits con segno, range da -32768 a 32767.
UINT_TYPE	USINT	32	Variabile word (UINT) 16 bits senza segno, range da 0 a 65535.
DWORD_TYPE	USINT	40	Variabile double word (DWORD) 32 bits senza segno, range da 0 a 4294967295.
DINT_TYPE	USINT	41	Variabile double word (DINT) 32 bits con segno, range da -2147483648 a 2147483647.
UDINT_TYPE	USINT	42	Variabile double word (UDINT) 32 bits senza segno, range da 0 a 4294967295.
REAL_TYPE	USINT	43	Variabile floating (REAL) 32 bits con segno, range da -3.40E+38 a +3.40E+38.
STRING_TYPE	USINT	50	Variabile stringa (STRING).

5.2 Task ID definition, identificatore di task PLC

Le task PLC sono identificate da un un valore, il valore è indicato con definizioni che si possono trovare nella cartella ***Task ID definition***.

Define	Type	Value	Description
ID_TASK_BOOT	USINT	0	Identifica la task di boot PLC. Questa task viene eseguita solo al primo loop di esecuzione programma utente.
ID_TASK_BACK	USINT	1	Identifica la task di background. Questa task è eseguita in background alle task slow e fast. Il tempo di loop di questa task non è fisso ma dipende dal carico di lavoro della CPU nella esecuzione delle altre tasks.
ID_TASK_SLOW	USINT	2	Identifica la task slow. Questa task è eseguita con un tempo di loop fisso definito con la funzione SysSetTaskLpTime. Di default il tempo è fissato a 10 mS.
ID_TASK_FAST	USINT	3	Identifica la task fast. Questa task è eseguita con un tempo di loop fisso definito con la funzione SysSetTaskLpTime. Di default il tempo è fissato a 1 mS.

5.3 TermIO definition, definizioni per terminale di I/O

Nella gestione dei terminali di I/O sono utilizzate delle definizioni che si possono trovare nella cartella ***TermIO definition***.

Define	Type	Value	Description
NULL	FILEP	0	Identifica un puntatore vuoto. Utilizzato come ritorno da alcune funzioni in caso di errore.
EOF	INT	-1	Identifica il fine file. Utilizzato come valore di ritorno da alcune funzioni in caso di fine file raggiunto.

5.4 FSeek origin definition, definizioni per seek su file

Nella gestione della ricerca su file sono utilizzate delle definizioni che si possono trovare nella cartella ***FSeek origin definition***.

Define	Type	Value	Description
ID_SEEK_SET	USINT	0	Inizio del file
ID_SEEK_CUR	USINT	1	Posizione corrente file
ID_SEEK_END	USINT	2	Fine del file

5.5 Serial mode definition, definizioni modo seriale

Nella gestione dei terminali di I/O sono utilizzate delle definizioni che si possono trovare nella cartella ***Serial mode definition***.

Define	Type	Value	Description
DTR_OFF	USINT	0	Valore membro DTRManagement del dato SYSSERIALMODE, indica segnale DTR sempre in condizione di off.
DTR_ON	USINT	1	Valore membro DTRManagement del dato SYSSERIALMODE, indica segnale DTR sempre in condizione di on.
DTR_AUTO_WO_TIMES	USINT	2	Valore membro DTRManagement del dato SYSSERIALMODE, indica segnale DTR in funzionamento automatico senza interposizione di tempi.
DTR_AUTO_W_TIMES	USINT	3	Valore membro DTRManagement del dato SYSSERIALMODE, indica segnale DTR in funzionamento automatico con interposizione di tempi.

5.6 CAN bit rate definition, definizioni bit rate CAN

Le definizioni dei valori di bit rate sul controller CAN si possono trovare nella cartella ***CAN bit rate definition***.

Define	Type	Value	Description
CAN_50KBIT	USINT	0	Bit rate 50 KBit
CAN_100KBIT	USINT	1	Bit rate 100 KBit
CAN_125KBIT	USINT	2	Bit rate 125 KBit
CAN_250KBIT	USINT	3	Bit rate 250 KBit
CAN_500KBIT	USINT	4	Bit rate 500 KBit
CAN_1MBIT	USINT	5	Bit rate 1 MBit

5.7 Digital input mode, definizioni modo acquisizione ingressi digitali

Le definizioni modo di acquisizione dei moduli di ingresso digitali si possono trovare nella cartella ***Digital input mode***.

Define	Type	Value	Description
DI_8_LL	USINT	1	Read 0-7 input mode (Debounced)
DI_8_L	USINT	2	Read 8-15 input mode (Debounced)
DI_8_M	USINT	3	Read 16-23 input mode (Debounced)
DI_8_MM	USINT	4	Read 24-31 input mode (Debounced)
DI_16_L	USINT	5	Read 0-15 input mode (Debounced)
DI_16_M	USINT	6	Read 16-31 input mode (Debounced)
DI_32	USINT	7	Read 0-31 input mode (Debounced)
DI_I_8_LL	USINT	11	Read 0-7 input mode (Immediate)
DI_I_8_L	USINT	12	Read 8-15 input mode (Immediate)
DI_I_8_M	USINT	13	Read 16-23 input mode (Immediate)
DI_I_8_MM	USINT	14	Read 24-31 input mode (Immediate)
DI_I_16_L	USINT	15	Read 0-15 input mode (Immediate)
DI_I_16_M	USINT	16	Read 16-31 input mode (Immediate)
DI_I_32	USINT	17	Read 0-31 input mode (Immediate)

5.8 Digital output mode, definizioni modo gestione uscite digitali

Le definizioni modo di gestione dei moduli di uscita digitali si possono trovare nella cartella ***Digital output mode***.

Define	Type	Value	Description
DO_8_LL	USINT	1	Write 0-7 output mode
DO_8_L	USINT	2	Write 8-15 output mode
DO_8_M	USINT	3	Write 16-23 output mode
DO_8_MM	USINT	4	Write 24-31 output mode
DO_16_L	USINT	5	Write 0-15 output mode
DO_16_M	USINT	6	Write 16-31 output mode
DO_32	USINT	7	Write 0-31 output mode

5.9 Analog to digital mode, definizioni modo acquisizione ingressi analogici

Le definizioni modo di acquisizione degli ingressi analogici si possono trovare nella cartella ***Analog to digital mode***.

Define	Type	Value	Description
AD_IDLE	USINT	0	Idle mode
AD_CURR_0_20_COMMON	USINT	3	Current from 0 to 20 mA (Common mode)
AD_CURR_0_20_DIFFER	USINT	6	Current from 0 to 20 mA (Differential mode)
AD_CURR_4_20_COMMON	USINT	4	Current from 4 to 20 mA (Common mode)
AD_CURR_4_20_DIFFER	USINT	13	Current from 4 to 20 mA (Differential mode)
AD_NI1000_DIFFER	USINT	12	Ni1000 sensor Celsius degree (Differential mode)
AD_NI1000_DIN_43760	USINT	48	Ni1000 DIN_43760 standard Celsius degree
AD_PT100_AMERICAN	USINT	33	Pt100 American standard Celsius degree
AD_PT100_DIFFER	USINT	10	Pt100 sensor Celsius degree (Differential mode)
AD_PT100_DIN_43760	USINT	32	Pt100 DIN_43760 standard Celsius degree
AD_PT100_IEC_60751	USINT	35	Pt100 IEC-60751 standard Celsius degree
AD_PT100_ITS_90	USINT	34	Pt100 ITS-90 standard Celsius degree
AD_PT1000_AMERICAN	USINT	41	Pt1000 American standard Celsius degree
AD_PT1000_DIFFER	USINT	11	Pt1000 sensor Celsius degree (Differential mode)
AD_PT1000_DIN_43760	USINT	40	Pt1000 DIN_43760 standard Celsius degree
AD_PT1000_IEC_60751	USINT	43	Pt1000 IEC-60751 standard Celsius degree
AD_PT1000_ITS_90	USINT	42	Pt1000 ITS-90 standard Celsius degree
AD_RESISTOR_300_OHM	USINT	110	Resistor up to 300 Ohm
AD_RESISTOR_5000_OHM	USINT	111	Resistor up to 5000 Ohm
AD_THERMOCOUPLE_B	USINT	64	Thermocouple B type Celsius degree
AD_THERMOCOUPLE_E	USINT	65	Thermocouple E type Celsius degree
AD_THERMOCOUPLE_J	USINT	66	Thermocouple J type Celsius degree
AD_THERMOCOUPLE_K	USINT	67	Thermocouple K type Celsius degree
AD_THERMOCOUPLE_N	USINT	68	Thermocouple N type Celsius degree
AD_THERMOCOUPLE_R	USINT	69	Thermocouple R type Celsius degree
AD_THERMOCOUPLE_S	USINT	70	Thermocouple S type Celsius degree
AD_THERMOCOUPLE_T	USINT	71	Thermocouple T type Celsius degree
AD_VIN_VREF_G_1	USINT	90	Voltage in/Voltage reference with gain 1
AD_VIN_VREF_G_128	USINT	97	Voltage in/Voltage reference with gain 128
AD_VIN_VREF_G_16	USINT	94	Voltage in/Voltage reference with gain 16
AD_VIN_VREF_G_2	USINT	91	Voltage in/Voltage reference with gain 2
AD_VIN_VREF_G_32	USINT	95	Voltage in/Voltage reference with gain 32
AD_VIN_VREF_G_4	USINT	92	Voltage in/Voltage reference with gain 4
AD_VIN_VREF_G_64	USINT	96	Voltage in/Voltage reference with gain 64
AD_VIN_VREF_G_8	USINT	93	Voltage in/Voltage reference with gain 8
AD_VOLT_0_1_COMMON	USINT	5	Voltage from 0 to 1 V (Common mode)
AD_VOLT_0_1_DIFFER	USINT	7	Voltage from 0 to 1 V (Differential mode)
AD_VOLT_0_10_COMMON	USINT	2	Voltage from 0 to 10 V (Common mode)
AD_VOLT_0_10_DIFFER	USINT	9	Voltage from 0 to 10 V (Differential mode)
AD_VOLT_0_125_COMMON	USINT	1	Voltage from 0 to 1.25 V (Common mode)
AD_VOLT_0_125_DIFFER	USINT	8	Voltage from 0 to 1.25 V (Differential mode)

5.10 Digital to analog mode, definizioni modo gestione uscite analogiche

Le definizioni modo di gestione uscite analogiche si possono trovare nella cartella ***Digital to analog mode***.

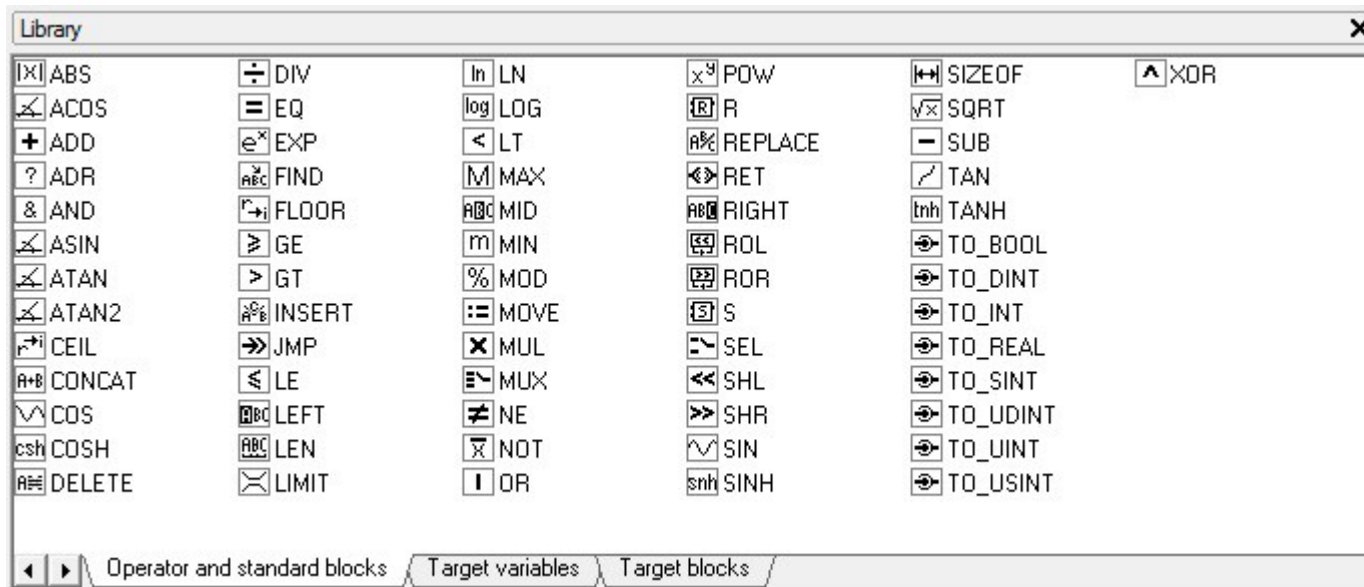
Define	Type	Value	Description
DA_CURR_0_20	USINT	5	Current from 0 to 20 mA
DA_CURR_4_20	USINT	6	Current from 4 to 20 mA
DA_VOLT_0_10	USINT	1	Voltage from 0 to 10 V
DA_VOLT_0_5	USINT	2	Voltage from 0 to 5 V
DA_VOLT_M10_10	USINT	3	Voltage from -10 to +10 V
DA_VOLT_M5_5	USINT	4	Voltage from -5 to +5 V

5.11 Spy mode, definizioni modo spionaggio dati

Le definizioni modo di spionaggio dati si possono trovare nella cartella ***Spy mode definition***.

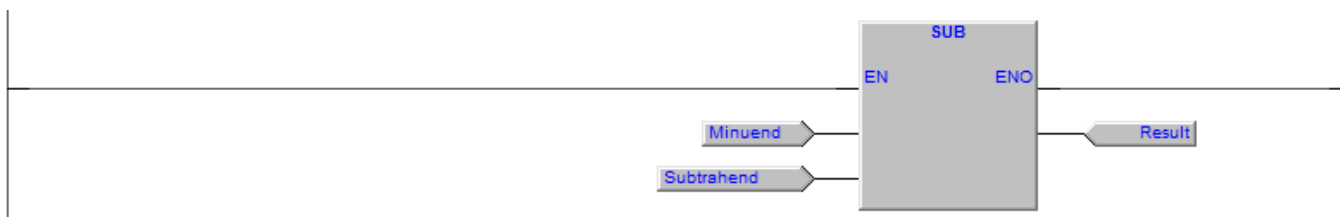
Define	Type	Value	Description
SPY_ASCII	USINT	1	Ascii mode (Example: 12:26:52(0000)|Tx|Messaggio 2)
SPY_ASCHEX	USINT	10	Extended ascii mode (Example: 12:28:47(0000)|Tx|M=000 [0D] [0A])
SPY_BINARY	USINT	100	Binary mode (Example: 12:26:52(0003)|Tx|00 00 00 00 00 00 00 00)

6 Funzioni definite da LogicLab

LogicLab supporta in modo nativo una serie di funzioni che possono essere utilizzata dai vari programmi scritti nei linguaggi supportati. Per utilizzare queste funzioni occorre dalla libreria nella sezione **Operator and standard blocks** attingere la funzione desiderata. Nei linguaggi testuali IL ed ST è anche possibile scrivere il nome della funzione esattamente così come è definita.

Ecco alcuni esempi di utilizzo nei vari linguaggi.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Minuend	UINT	Auto	No	0	..	Minuend
2	Subtrahend	UINT	Auto	No	0	..	Subtrahend
3	Result	UINT	Auto	No	0	..	Result

Esempio LD

Esempio IL

```
LD Minuend
SUB Subtrahend
ST Result
```

Esempio ST

```
Result:=SUB(Minuend, Subtrahend); (* Subtraction using SUBB *)
Result:Minuend-Subtrahend; (* Subtraction as before *)
```

6.1 Funzioni matematiche e trigonometriche

LogicLab supporta tutte le funzioni matematiche richieste dalla norma CEI EN 61131-3 nella Parte 3: Linguaggi di programmazione. Riporto di seguito un elenco delle funzioni supportate ed un esempio di utilizzo in linguaggio ST.

ABS

ANY_NUM ABS(ANY_NUM)
Calcola il valore assoluto di un numero.

```
Result:=ABS(-10.5); (* Result is 10.5 *)
```

SQRT

ANY_REAL SQRT(ANY_REAL)
Calcola la radice quadrata di un numero.

```
Result:=SQRT(9.0); (* Result is 3.0 *)
```

LN

ANY_REAL LN(ANY_REAL)
Calcola logaritmo naturale (Base "e" 2,71828) di un numero.

```
Result:=LN(10.0); (* Result is 2.30259 *)
```

LOG

ANY_REAL LOG(ANY_REAL)
Calcola logaritmo (Base "10") di un numero.

```
Result:=LOG(10.0); (* Result is 1.0 *)
```

EXP

ANY_REAL EXP(ANY_REAL)
Calcola numero elevato "e" 2,71828.

```
Result:=EXP(1.0); (* Result is 2.71828 *)
```

SIN

ANY_REAL SIN(ANY_REAL)
Calcola il seno di un angolo in radianti.

```
Result:=SIN(1.57); (* Angle is 90°, Result is 1.0 *)
```

COS

ANY_REAL COS(ANY_REAL)
Calcola il coseno di un angolo in radianti.

```
Result:=COS(3.1416); (* Angle is 180°, Result is -1.0 *)
```

TAN

ANY_REAL TAN(ANY_REAL)
Calcola la tangente di un angolo in radianti.

```
Result:=TAN(0.7854); (* Angle is 45°, Result is 1.0 *)
```

ASIN

ANY_REAL ASIN(ANY_REAL)
Calcola l'arcoseno di un angolo in radianti.

```
Result:=ASIN(1.0); (* Result is: 1.5708 *)
```

ACOS

ANY_REAL ACOS(ANY_REAL)
Calcola l'arcocoseno di un angolo in radianti.

```
Result:=ACOS(-1.0); (* Result is 3.14159 *)
```

ATAN

ANY_REAL ATAN(ANY_REAL)
Calcola l'arcotangente di un angolo in radianti.

```
Result:=ATAN(1.0); (* Result is 3.14159 *)
```

ADD

ANY_NUM ADD(ANY_NUM, ANY_NUM)
Esegue somma tra due numeri.

```
Result:=ADD(1.0, 2.0); (* Result is 3.0 *)
```

MUL

ANY_NUM MUL(ANY_NUM, ANY_NUM)
Esegue moltiplicazione tra due numeri.

```
Result:=MUL(1.0, 2.0); (* Result is 2.0 *)
```

SUB

ANY_NUM SUB(ANY_NUM, ANY_NUM)
Esegue sottrazione tra due numeri.

```
Result:=SUB(2.0, 1.0); (* Result is 1.0 *)
```

DIV **ANY_NUM DIV(ANY_NUM, ANY_NUM)**

Esegue divisione tra due numeri.

```
Result:=DIV(2.0, 1.0); (* Result is 2.0 *)
```

Considerazione a parte và fatta per l'operatore modulo **MOD**. L'aritmetica modulare si applica ai soli numeri interi, nel quale i numeri "si avvolgono su se stessi" ogni volta che raggiungono i multipli di un determinato numero n, detto modulo.

L'operazione Result:=x MOD 10, darà come risultato valori compresi tra 0 e 10 per qualsiasi valore assuma x. L'operazione Result:=x MOD 1000, darà come risultato valori compresi tra 0 e 1000 per qualsiasi valore assuma x.

6.2 Funzioni stringa

LogicLab supporta tutte le funzioni di gestione stringa richieste dalla norma CEI EN 61131-3 nella Parte 3: Linguaggi di programmazione. Riporto di seguito un elenco delle funzioni supportate ed un esempio di utilizzo in linguaggio ST.

CONCAT

STRING CONCAT(STRING S0, STRING S1)

Concatena le stringhe S0 ed S1.

```
AString:='Hello ';
BString:='World !';
CString:=CONCAT(AString, BString); (* CString is 'Hello World !' *)
CString:=CONCAT(AString, 'World !'); (* CString is 'Hello World !' *)
```

DELETE

STRING DELETE(STRING IN, ANY_INT L, ANY_INT P)

Cancella dalla stringa IN, L caratteri partendo dalla posizione P a scendere.

```
AString:='Hello World !';
BString:=DELETE(AString, 2, 3);  (* BString is 'Heo World !' *)
```

FIND

UINT FIND(STRING S0, STRING S1)

Cerca la posizione della prima occorrenza di S1 in S0. Se non trovata torna 0.

```
AString:='Hello World !';
i:=FIND(AString, 'World'); (* Result is: 7 *)
j:=FIND('Hello World World !', 'World'); (* Result is 7 *)
k:=FIND('World', 'Hello'); (* Result is: 0 *)
```

INSERT

STRING INSERT(STRING S0, STRING S1, ANY_INT P)

Inserisce nella stringa S0 la stringa S1 partendo dalla posizione P.

```
AString:='Hello everybody';
BString:='World !';
CString:=INSERT(AString, BString, 6); (* CString is 'Hello World !'*)
```

LEFT

STRING LEFT(STRING IN, ANY_INT L)

Ritorna gli L caratteri più a sinistra della stringa IN.

```
AString:='Hello World !';
BString:=LEFT(AString, 7); (* BString is 'Hello W'
```

LEN

UINT LEN(STRING IN)

Ritorna la lunghezza (Numero di caratteri) della stringa IN.

```
AString:='Hello World !';
i:=LEN(AString); (* i is: 13 *)
```

MID

STRING MID(STRING IN, ANY_INT L, ANY_INT P)

Ritorna gli L caratteri della stringa IN partendo dal carattere in posizione P.

```
AString:='Hello World !';
BString:=MID(AString, 5, 7); (* BString is 'World' *)
```

REPLACE

STRING REPLACE(STRING S0, STRING S1, ANY_INT L, ANY_INT P)

Rimpiazza L caratteri della stringa S0 con la stringa S1 a partire dalla posizione P.

```
AString:='Hello World !';
BString:=REPLACE(AString, 'to you ', 5, 7);
```

RIGHT

STRING RIGHT(STRING IN, ANY_INT L)

Ritorna gli L caratteri più a destra della stringa IN.

```
AString:='Hello World !';
BString:=RIGHT(AString, 7); (* BString is 'World !' *)
```

7 Funzioni ed FB

Funzioni

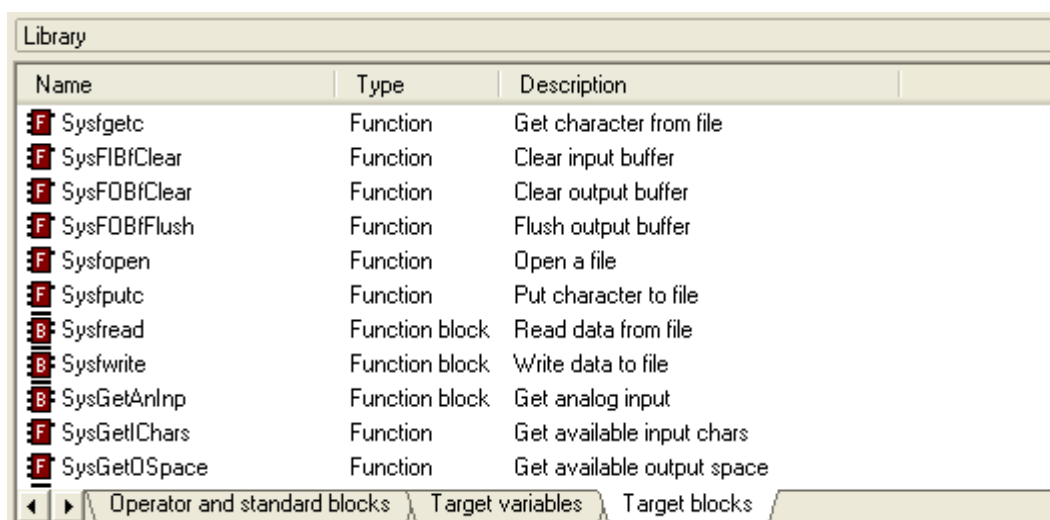
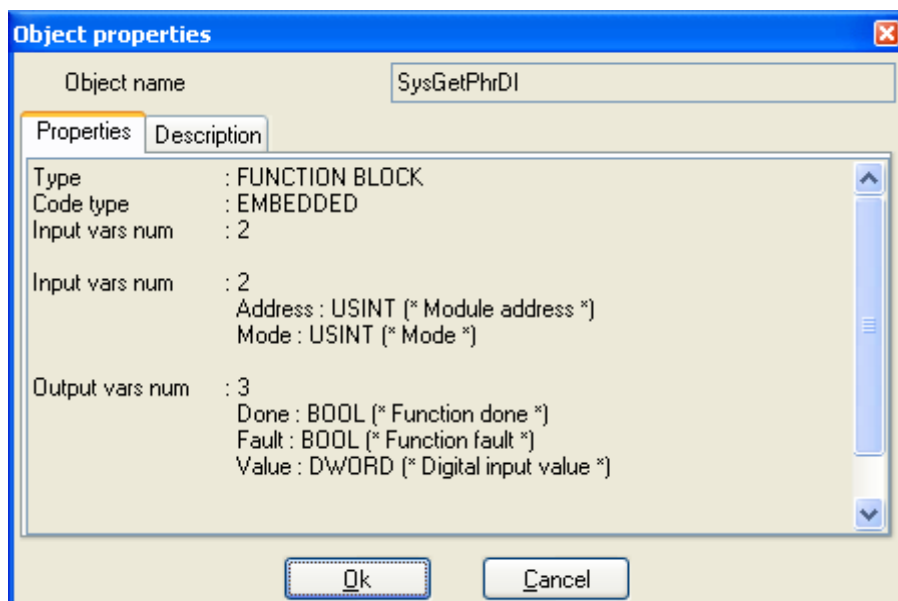
Le funzioni hanno numero di variabili in ingresso variabile e sempre una sola variabile in uscita. Per utilizzarle basta inserirle nei programmi LD ed FBD e connetterle alle variabili. Nei programmi IL devono essere chiamate con l'istruzione CAL, nei programmi ST basta indicarne il nome per essere eseguite.

Function Blocks

Le FB a differenza delle funzioni allocano nel programma una variabile che contiene tutte le variabili di input e di output gestite dal blocco funzione. Per utilizzarle basta inserirle nei programmi LD ed FBD e connetterle alle variabili. Nei programmi IL devono essere chiamate con l'istruzione CAL, nei programmi ST basta indicarne il nome per essere eseguite.

7.0.1 Funzioni ed FB embedded

Con l'ambiente di sviluppo LogicLab vengono fornite funzioni e blocchi funzione (FB) embedded che permettono di accedere alle risorse hardware e software del sistema ***Slim line***. Le funzioni e le FB embedded sono visualizzate da LogicLab nella finestra delle librerie.

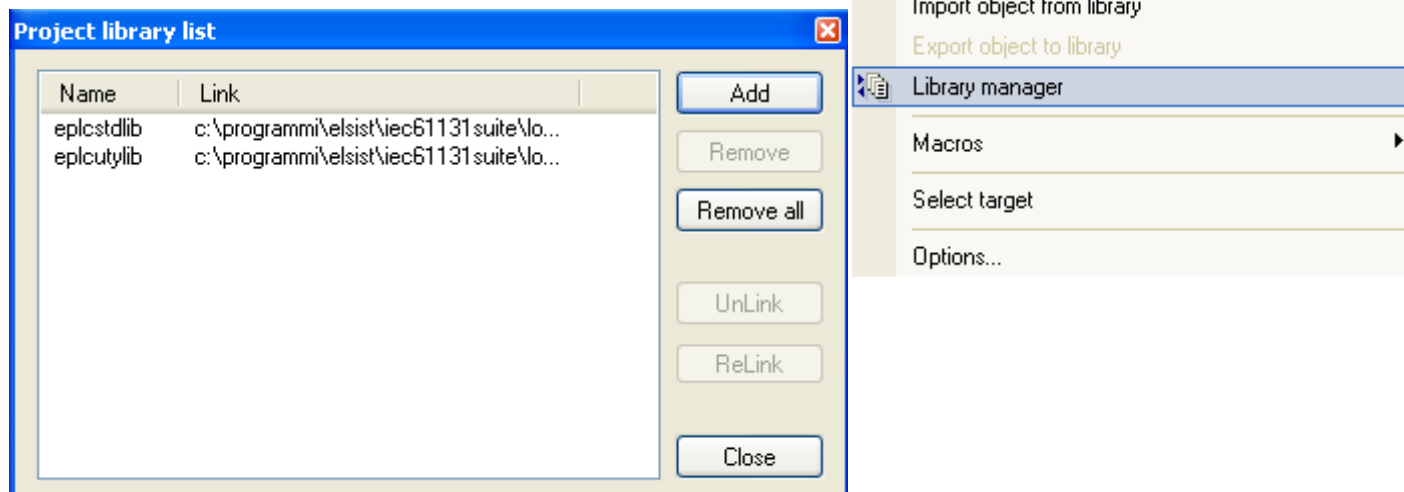
Se la finestra non è visualizzata, occorre abilitarne la visualizzazione dalla voce di menù ***View*** → ***Tool windows*** → ***Library***. Attivando il Tab ***Target blocks*** verrà visualizzato un elenco con tutte le funzioni (Indicate con ***F***) ed i blocchi funzione (Indicati con ***B***) embedded.

Agendo con il tasto destro del mouse su ogni singola funzione o blocco funzione è possibile visualizzarne la finestra delle proprietà in cui sono indicate le variabili in ingresso ed il ritorno delle funzioni, mentre per i blocchi funzioni sono indicate le variabili in ingresso ed in uscita, cosi come nella figura a lato.

7.0.2 Librerie

In aggiunta alle funzioni ed FB embedded sono fornite delle librerie che contengono una serie di funzioni e blocchi funzioni che possono essere utilizzati nel proprio programma. Le librerie fornite con LogicLab si trovano nella directory di installazione programma ***Programmi\Elsist\IEC61131Suite\LogicLab2p0\Libraries***, ma è possibile anche utilizzare librerie fornite successivamente o di cui si è eseguito il download dal sito. Esistono due possibilità per utilizzare le librerie:

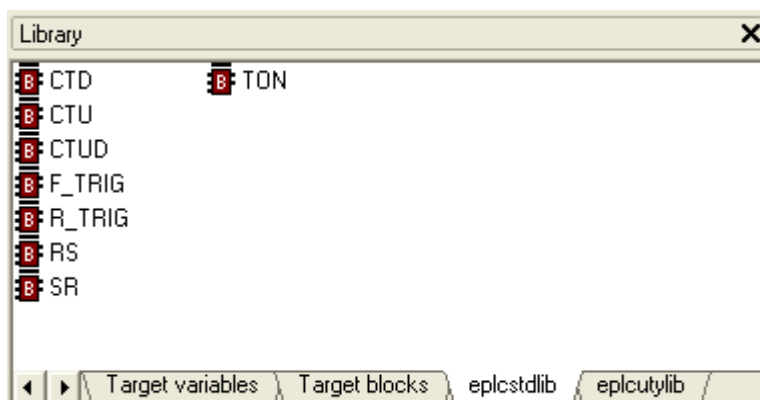
Import libreria: In questo modo vengono importati nel proprio programma tutti gli oggetti presenti nella libreria, gli oggetti possono così essere utilizzati nel programma. Questa è una buona soluzione. La controindicazione è quella di aumentare la dimensione del file di progetto LogicLab (*.ppjs), in quanto deve contenere oltre al proprio programma anche tutti gli oggetti della libreria importata. ***Il programma eseguibile generato conterrà comunque solo gli oggetti utilizzati***.

Import oggetti: In questo modo è possibile importare da una libreria solo gli oggetti (Funzioni, FB, ecc) che interessano, i quali diverranno parte integrante del proprio progetto.

7.0.3 Import libreria

Con questa modalità vengono importati tutti gli oggetti presenti nella libreria. Per importare nel proprio programma l'intera libreria dal menù selezionare la voce ***Project*** → ***Library manager*** si aprirà una finestra come quella sotto riportata.

Agendo sul tasto ***Add*** si aprirà una finestra di browser del disco. Scegliere la directory dove si trova la libreria, e selezionare i files di libreria da importare.

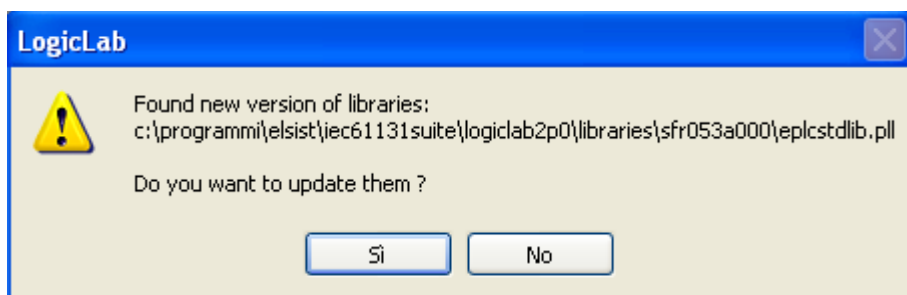
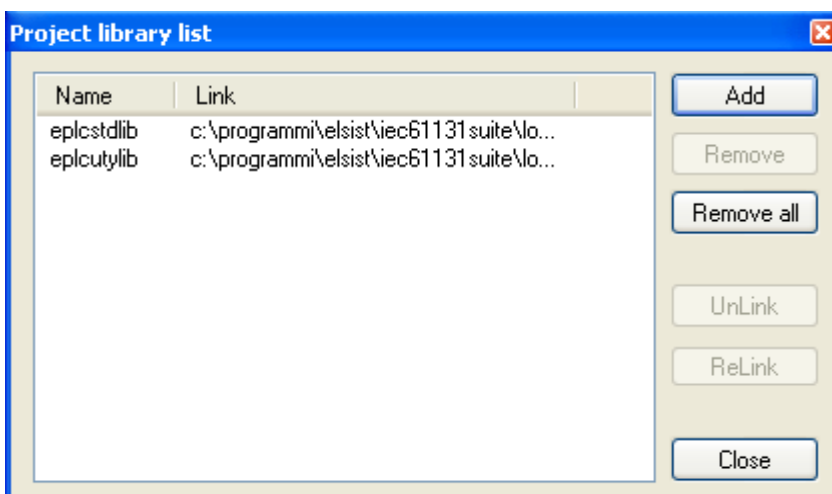
Agendo sul tasto ***Close***, nella finestra ***Library*** di LogicLab (***Ctrl-L***) verranno visualizzati dei tabs aggiuntivi, uno per ogni libreria importata.

Basta trascinare l'oggetto desiderato nel proprio progetto per poterlo utilizzare.

7.0.4 Importazione libreria

Eseguendo l'import delle librerie nel proprio progetto come indicato al capitolo precedente tutti gli oggetti della libreria importata vengono trasferiti nel proprio file di progetto (*.ppjs), ma viene comunque mantenuto un link alla libreria di origine come si vede dalla finestra a lato.

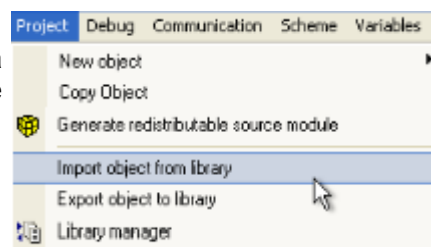
Questo permette nel caso la libreria sorgente venga modificata con una versione più recente di effettuare l'aggiornamento automatico della nuova libreria nel proprio progetto.

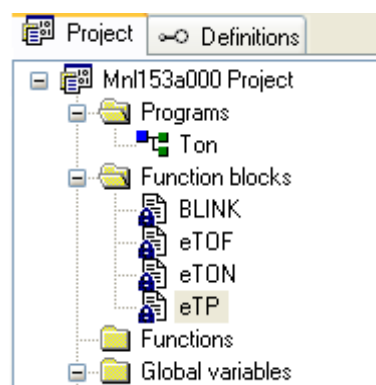
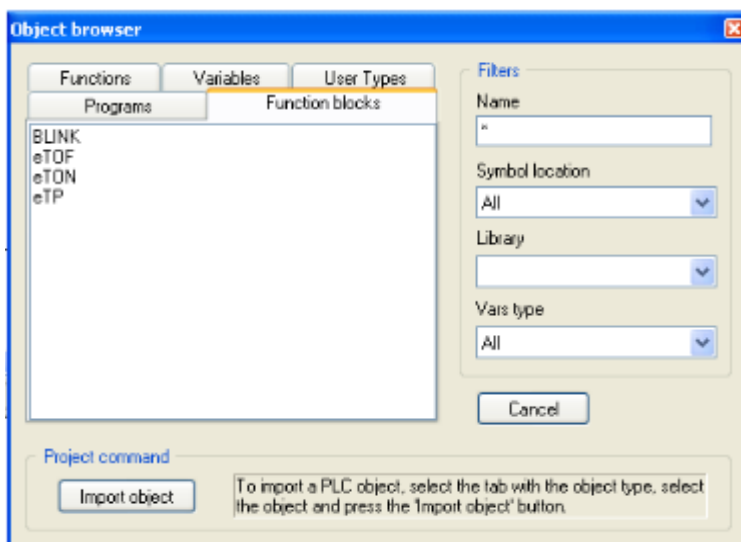
Se la libreria sorgente non è più presente oppure è stata spostata dalla posizione da dove è stata importata, LogicLab non eseguirà più il controllo senza segnalare errori.

Tramite il menù ***Project → Library manager*** che apre la finestra a lato, come si vede, è possibile selezionare le varie librerie e con il tasto ***UnLink*** rimuovere il link oppure con il tasto ***ReLink*** eseguire un link alla nuova posizione dove si trova la libreria.

Aprendo il progetto, LogicLab controlla tutte le librerie importate e nel caso una o più librerie sorgenti siano più recenti delle versioni importate viene visualizzato un messaggio di avvertimento che chiede conferma se eseguire oppure no l'aggiornamento delle librerie.

Eseguendo l'aggiornamento tutti gli oggetti della libreria importata presenti nel proprio progetto vengono sovrascritti con gli oggetti presenti nella libreria sorgente ed eventuali nuovi oggetti sono automaticamente importati.

7.0.5 Importazione oggetti da libreria

Per importare nel proprio programma oggetti dalle librerie occorre nel menù selezionare la voce ***Project → Import object from library***. Si aprirà una finestra di browser del disco. Scegliere la directory dove si trova la libreria, e selezionare il file di libreria da cui si desidera importare gli oggetti.

Si aprirà la finestra di ***Object browser*** che permetterà di visualizzare tutti gli oggetti presenti nella libreria. Selezionando i vari tabulatori presenti è possibile visualizzare tutti gli oggetti della libreria ordinati per nome.

Con un click del mouse si evidenzia l'oggetto o gli oggetti desiderati. Con il tasto ***Import object*** gli oggetti selezionati verranno inclusi nel programma.

Come si nota dalla foto a destra alcuni oggetti appaiono visualizzati con un simbolo di lucchetto, questo sta ad indicare che sono oggetti protetti, cioè non possono essere modificati. Una volta importati nel proprio programma gli oggetti resteranno inclusi nel programma stesso e sarà possibile utilizzarli su qualsiasi PC anche se non si dispone della libreria originale.

7.0.6 Considerazioni su link a libreria e su import oggetti

Come visto nei paragrafi precedenti per utilizzare funzioni e/o blocchi funzione di libreria è possibile usare due diversi metodi, importare solo l'oggetto desiderato oppure tutta la libreria nel proprio progetto.

In entrambi i casi l'oggetto verrà incluso nel proprio progetto, in questo modo si è sicuri che anche nel futuro con versioni successive di libreria sarà sempre possibile ricompilare il progetto utilizzando l'oggetto con il quale si era sviluppato e testato.

Nel caso si desideri sostituire l'oggetto con una versione più recente dello stesso si userà un diverso approccio in funzione del fatto che l'oggetto sia presente in una libreria collegata oppure sia stato importato.

Libreria collegata

Come visto precedentemente, le librerie collegate mantengono un riferimento alla libreria di origine, nel percorso di memorizzazione nella distribuzione di LogicLab le librerie sono incluse in directories il cui nome rappresenta la versione. In questo modo potranno essere distribuite versioni successive di libreria, ma il progetto alla sua riapertura farà sempre il controllo con la versione originale senza eseguire l'aggiornamento automatico.

Per effettuare l'aggiornamento di un oggetto di una libreria collegata occorre eseguire un ***ReLink*** alla nuova versione della libreria. ***Attenzione!*** Questa operazione aggiornerà tutti gli oggetti presenti nella libreria.

Oggetto importato

Nel caso di oggetto importato, per effettuare l'aggiornamento, basterà rimuovere l'oggetto attuale dal progetto ed eseguire un import dello stesso oggetto dalla nuova versione della libreria.

Conclusioni

In generale si consiglia di non eseguire il collegamento della libreria ma di includere i singoli oggetti nel proprio progetto, questo permette una più semplice gestione degli aggiornamenti.

Alcune librerie contengono una serie di oggetti (Funzioni e blocchi funzione) che sono di vasto impiego, in questo caso è consigliabile sempre collegare queste librerie. Ecco l'elenco delle librerie che si consiglia di collegare al progetto:

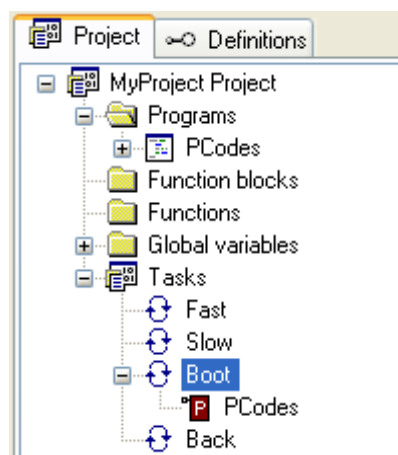
Libreria	Codice	Descrizione
ePLCStdLib	SFR053*000	Libreria standard IEC61131, contiene funzioni e blocchi funzione definiti dalla normativa IEC61131 e non presenti nella libreria embedded del prodotto.
ePLCAuxLib	SFR058*000	Libreria ausiliaria, contiene funzioni e blocchi funzione di varia utilità.

7.0.7 Protezione funzioni e blocchi funzione

Alcune funzioni e/o blocchi funzione di libreria possono essere protetti da un codice che deve essere ordinato separatamente. Per abilitarne l'esecuzione occorre sbloccarle definendone il codice (Stringa alfabetica di 18 caratteri) con la funzione ***SysPCodeAccept***.

La funzione deve essere eseguita una sola volta passando il codice di protezione, se il codice è corretto la funzione ritorna ***TRUE*** e la relativa funzione sarà sprotetta fino al prossimo riavvio del programma. E' possibile eseguire più chiamate alla funzione una per ogni codice di protezione da definire.

Il consiglio è di inserire le varie chiamate alla funzione in un programma che verrà eseguito nella task di boot quindi prima di ogni chiamata ad altri programmi, garantendo lo sblocco delle funzioni desiderate.

A lato si può vedere come in un progetto il programma di definizione codici di protezione ***PCodes*** sia definito nella esecuzione della task di boot. Di seguito riportiamo il codice sorgente del programma ***PCodes*** realizzato in linguaggio ST. Naturalmente i codici riportati sono di fantasia pertanto se eseguito la funzione ***SysPCodeAccept*** ritornerà sempre ***FALSE***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	CodesOk	BOOL	Auto	[0..2]	3(0)	..	Protection codes ok

Programma ST

```
(* Check the protection codes. *)

    CodesOk[0]:=SysPCodeAccept('abcdefghijklmnopqr'); (* Protection code ok (Function 1) *)
    CodesOk[1]:=SysPCodeAccept('rqponmlkjihgfedcba'); (* Protection code ok (Function 2) *)
    CodesOk[2]:=SysPCodeAccept('abcdefghiihgfedcba'); (* Protection code ok (Function 3) *)

(* [End of file] *)
```

Normalmente le funzioni ed i blocchi funzioni protetti possono funzionare in modo demo per un certo periodo di tempo dalla loro prima esecuzione dopo l'accensione del sistema. Terminato il tempo di prova termina il funzionamento e viene generato un errore che è rilevabile con la funzione ***SysGetLastError***.

7.1 Funzioni ed FB per gestione Flip/Flop

7.1.1 F_TRIG, Falling edge trigger

Type	Library
FB	ePLCStdLib_B000

Questo blocco funzione attiva l'uscita ***Q*** per un loop di programma sul fronte di disattivazione dell'ingresso di clock ***CLK***.

CLK (BOOL) Clock, sul fronte di disattivazione del segnale, viene attivata l'uscita Q per un loop di programma.

Q (BOOL) Uscita, attiva per un loop di programma sul fronte di disattivazione dell'ingresso di clock CLK.

Esempi

Sul fronte di disattivazione dell'ingresso digitale ***Di00M00*** viene attivata per un loop di programma l'uscita digitale ***Do00M00***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBF_TRIG	F_TRIG	Auto	No	0	..	F_TRIG (Falling edge trigger function block)

Esempio LD *(PTP115A000, F_TRIG_LD)*

Esempio IL

```
CAL FBF_TRIG (* Call the F_TRIG function block *)

LD Di00M00
ST FBF_TRIG.CLK (* Transfer the digital input to the function block clock *)

LD FBF_TRIG.Q
ST Do00M00 (* On the falling edge of digital input the output is set *)
```

Esempio ST

```
FBF_TRIG(); (* Call the F_TRIG function block *)

FBF_TRIG.CLK:=Di00M00; (* Transfer the digital input to the function block clock *)
Do00M00:=FBF_TRIG.Q; (* On the falling edge of digital input the output is set *)
```

7.1.2 R_TRIG, Raising edge trigger

Type	Library
FB	ePLCStdLib_B000

Questo blocco funzione attiva l'uscita ***Q*** per un loop di programma sul fronte di attivazione dell'ingresso di clock ***CLK***.

CLK (BOOL) Clock, sul fronte di attivazione del segnale, viene attivata l'uscita Q per un loop di programma.

Q (BOOL) Uscita, attiva per un loop di programma sul fronte di attivazione dell'ingresso di clock CLK.

Esempi

Sul fronte di attivazione dell'ingresso digitale ***Di00M00*** viene attivata per un loop di programma l'uscita digitale ***Do00M00***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBR_TRIG	R_TRIG	Auto	No	0	..	R_TRIG (Raising edge trigger function block)

Esempio LD *(PTP115A100, R_TRIG_LD)*

FB_R_TRIG
R_TRIG
Di00M00
CLK
Q
Do00M00

Esempio IL

```
CAL FBR_TRIG (* Call the R_TRIG function block *)

LD Di00M00
ST FBR_TRIG.CLK (* Transfer the digital input to the function block clock *)

LD FBR_TRIG.Q
ST Do00M00 (* On the raising edge of digital input the output is set *)
```

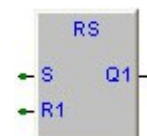
Esempio ST

```
FBR_TRIG(); (* Call the R_TRIG function block *)

FBR_TRIG.CLK:=Di00M00; (* On the raising edge of digital input the counter count up *)
Do00M00:=FBR_TRIG.Q; (* On the raising edge of digital input the output is set *)
```

7.1.3 RS, Reset/Set flip flop

Type	Library
FB	ePLCStdLib_B000

Questo blocco funzione su attivazione del comando di set ***S*** attiva l'uscita ***Q1*** che rimane attiva anche quando il comando viene disattivato. Per disattivare l'uscita occorre attivare il comando di reset ***R1***.

Il comando di reset R1 è prioritario sul comando di set S.

S (BOOL) Set, su attivazione del segnale, viene attivata l'uscita Q1 che rimane attiva anche quando il comando viene disattivato.

R1 (BOOL) Reset, su attivazione del segnale, viene disattivata l'uscita Q1 è prioritario sul comando di set S.

Q1 (BOOL) Uscita, si attiva e disattiva in funzione dei comandi di S set e R1 reset.

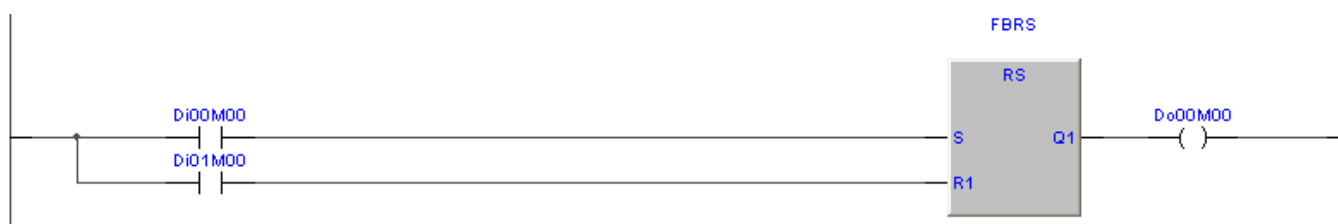
Esempi

Sulla attivazione dell'ingresso digitale ***Di00M00*** viene attivata l'uscita digitale ***Do00M00*** che rimane attiva anche quando l'ingresso digitale ***Di00M00*** viene disattivato. Per disattivare l'uscita digitale ***Do00M00*** occorre attivare l'ingresso digitale ***Di01M00***.

Nota! L'ingresso digitale ***Di01M00*** ha la priorità sull'ingresso digitale ***Di00M00***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBRS	RS	Auto	No	0	..	RS (Reset/Set function block)

Esempio LD *(PTP115A100, RS_LD)*

Esempio IL

```
CAL FBRS (* Call the RSG function block *)

LD Di00M00
ST FBRS.S (* Transfer the digital input to the set command *)

LD Di01M00
ST FBRS.R1 (* Transfer the digital input to the reset command *)

LD FBRS.Q1
ST Do00M00 (* The function block output is copied to digital output *)
```

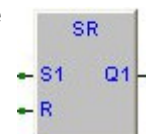
Esempio ST

```
FBRS(); (* Call the RS function block *)

FBRS.S:=Di00M00; (* Transfer the digital input to the set command *)
FBRS.R1:=Di01M00; (* Transfer the digital input to the reset command *)
Do00M00:=FBRS.Q1; (* The function block output is copied to digital output *)
```

7.1.4 SR, Set/Reset flip flop

Type	Library
FB	ePLCStdLib_B000

Questo blocco funzione su attivazione del comando di set ***S1*** attiva l'uscita ***Q1*** che rimane attiva anche quando il comando di set viene disattivato. Per disattivare l'uscita occorre attivare il comando di reset ***R***.

Il comando di set S1 è prioritario sul comando di reset R.

S1 (BOOL) Set, su attivazione del segnale, viene attivata l'uscita Q1 che rimane attiva anche quando il comando viene disattivato. Il comando è prioritario sul comando di reset R.

R (BOOL) Reset, su attivazione del segnale, viene disattivata l'uscita Q1.

Q1 (BOOL) Uscita, si attiva e disattiva in funzione dei comandi di S set e R reset.

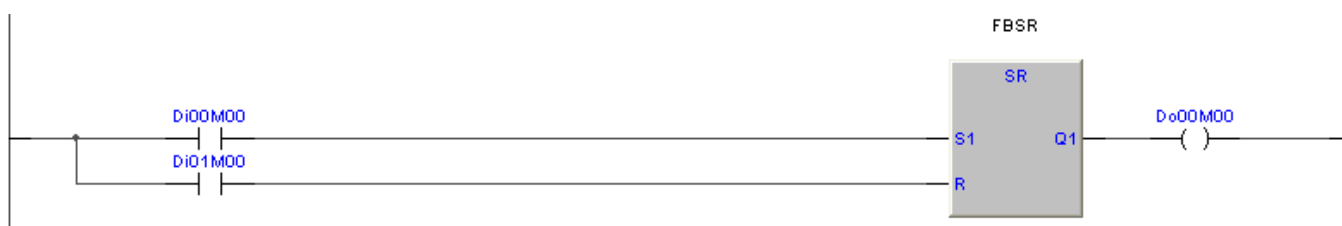
Esempi

Sulla attivazione dell'ingresso digitale ***Di00M00*** viene attivata l'uscita digitale ***Do00M00*** che rimane attiva anche quando l'ingresso digitale ***Di00M00*** viene disattivato. Per disattivare l'uscita digitale ***Do00M00*** occorre attivare l'ingresso digitale ***Di01M00***.

Nota! L'ingresso digitale ***Di00M00*** ha la priorità sull'ingresso digitale ***Di01M00***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBSR	SR	Auto	No	0	..	SR (Set/Reset function block)

Esempio LD *(PTP115A100, SR_LD)*

Esempio IL

```
CAL FBSR (* Call the SR function block *)

LD Di00M00
ST FBRS.S1 (* Transfer the digital input to the set command *)

LD Di01M00
ST FBRS.R (* Transfer the digital input to the reset command *)

LD FBRS.Q1
ST Do00M00 (* The function block output is copied to digital output *)
```

Esempio ST

```
FBSR(); (* Call the SR function block *)

FBSR.S1:=Di00M00; (* Transfer the digital input to the set command *)
FBSR.R:=Di01M00; (* Transfer the digital input to the reset command *)
Do00M00:=FBSR.Q1; (* The function block output is copied to digital output *)
```

7.2 Funzioni ed FB per gestione timers

7.2.1 eTOF, Timer Off

Type	Library
FB	ePLCStdLib_B000

Questo blocco funzione esegue la temporizzazione sulla disattivazione, attivando l'ingresso ***IN*** l'uscita ***Q*** si attiva immediatamente ed il tempo in uscita ***ET*** si azzera. Disattivando l'ingresso IN inizia il conteggio e dopo il tempo definito ***PT*** espresso in mS, si disattiva l'uscita ***Q***. Sulla uscita ***ET*** viene ritornato il tempo trascorso dalla disattivazione dell'ingresso espresso in mS.

IN (BOOL) Ingresso timer, attivandolo l'uscita Q si attiva immediatamente ed il tempo in uscita ET si azzera. Disattivandolo inizia il conteggio e dopo il tempo definito in PT, si disattiva l'uscita Q.

PT (UDINT) Preset tempo, definisce il tempo di ritardo dalla disattivazione dell'ingresso IN alla disattivazione dell'uscita Q, espresso in mS.

Q (BOOL) Uscita timer, si attiva su attivazione ingresso IN, e si disattiva dopo il tempo definito in PT dalla disattivazione dell'ingresso IN.

ET (UDINT) Tempo timer, si azzera su attivazione ingresso IN ed inizia conteggio da disattivazione ingresso IN. Raggiunto tempo impostato in PT si arresta conteggio, espresso in mS.

Esempi

Il timer è presettato a 1 secondo (1000 mS). Attivando l'ingresso digitale ***Di00M00*** si attiva immediatamente l'uscita digitale ***Do00M00*** ed il valore di tempo nella variabile ***VarOut*** è azzerato.

Disattivando l''ingresso digitale ***Di00M00*** il timer inizia il conteggio del tempo, il valore di tempo trascorso dalla disattivazione è trasferito nella variabile ***VarOut***. Trascorso il tempo l'uscita digitale ***Do00M00*** si disattiva.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBeTOF	eTOF	Auto	No	0	..	eTOF (Timer Off function block)
2	OutValue	UDINT	Auto	No	0	..	Output value

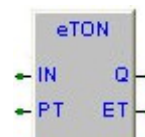
Esempio LD *(PTP115A100, eTOF_LD)*

Esempio IL

```
CAL FBeTOF (* Call the eTOF function block *)

LD  Di00M00
ST  FBeTOF.IN (* Transfer the digital input to timer input *)

LD  1000
ST  FBeTOF.PT (* Set the delay time *)

LD  FBeTOF.Q
ST  Do00M00 (* When time is passed the digital output is set *)

LD  FBeTOF.ET
ST  OutValue (* The counting time is copied to variable *)
```

7.2.2 eTON, Timer On

Type	Library
FB	ePLCStdLib_B000

Questo blocco funzione esegue la temporizzazione sulla attivazione, attivando l'ingresso ***IN*** inizia il conteggio e dopo il tempo definito ***PT*** espresso in mS, si attiva l'uscita ***Q***. Sulla uscita ***ET*** viene ritornato il tempo trascorso dalla attivazione dell'ingresso espresso in mS. Disattivando l'ingresso ***IN*** l'uscita ***Q*** si disattiva istantaneamente ed il valore di tempo su uscita ***ET*** si azzera.

IN (BOOL) Ingresso timer, attivandolo inizia il conteggio e dopo il tempo definito in PT, si attiva l'uscita Q. Disattivandolo l'uscita Q si disattiva immediatamente ed il tempo in uscita ET si azzera.

PT (UDINT) Preset tempo, definisce il tempo di ritardo dalla attivazione dell'ingresso IN alla attivazione dell'uscita Q, espresso in mS.

Q (BOOL) Uscita timer, si attiva dopo il tempo definito in PT dalla attivazione dell'ingresso IN e si disattiva su disattivazione ingresso IN.

ET (UDINT) Tempo timer, inizia conteggio da attivazione ingresso IN, raggiunto tempo impostato in PT si arresta conteggio. Si azzera su disattivazione ingresso IN, espresso in mS.

Esempi

Su attivazione dell'ingresso digitale ***Di00M00*** dopo 1 S (1000 mS) viene attivata l'uscita digitale ***Do00M00***. Disattivando l'ingresso digitale ***Di00M00*** l'uscita digitale ***Do00M00*** si disattiva immediatamente.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBeTON	eTON	Auto	No	0	..	eTON (Timer On function block)
2	OutValue	UDINT	Auto	No	0	..	Output value

Esempio LD *(PTP115A100, eTON_LD)*

Esempio IL

```
CAL FBeTON (* Call the eTON function block *)
LD  Di00M00
ST  FBeTON.IN (* Transfer the digital input to timer input *)

LD  1000
ST  FBeTON.PT (* Set the delay time *)

LD  FBeTON.Q
ST  Do00M00 (* When time is passed the digital output is set *)

LD  FBeTON.ET
ST  OutValue (* The counting time is copied to variable *)
```

Esempio ST

```
FBeTON(PT:=1000); (* Call the eTON function block *)

FBeTON.IN:=Di00M00; (* Transfer the digital input to timer input *)
OutValue:=FBeTON.ET; (* The counting time is copied to variable *)
Do00M00:=F_eTON.Q; (* When time is passed the digital output is set *)
```

7.2.3 eTP, Timer pulse

Type	Library
FB	ePLCStdLib_B000

Questo blocco funzione esegue la temporizzazione su impulso di attivazione, attivando l'ingresso ***IN*** l'uscita ***Q*** si attiva, in uscita ***ET*** è ritornato il tempo trascorso (in mS) dall'impulso di attivazione. Raggiunto il tempo impostato ***PT*** (in mS), indipendentemente dallo stato dell'ingresso ***IN***, l'uscita ***Q*** si azzera, mentre il tempo in uscita su ***ET*** si azzera solo se ingresso ***IN*** non è più attivo.

IN (BOOL) Ingresso timer, attivandolo si attiva l'uscita Q ed inizia il conteggio, dopo il tempo definito in PT indipendentemente dallo stato dell'ingresso IN, l'uscita Q si azzera.

PT (UDINT) Preset tempo, definisce il tempo di attivazione dell'uscita Q, espresso in mS.

Q (BOOL) Uscita timer, si attiva all'attivazione dell'ingresso IN per il tempo definito in PT.

ET (UDINT) Tempo timer, inizia conteggio da attivazione ingresso IN, raggiunto tempo impostato in PT si arresta conteggio, espresso in mS.

Esempi

Il timer è presettato a 5 secondi (5000 mS). Attivando l'ingresso digitale ***Di00M00*** si attiva immediatamente l'uscita digitale ***Do00M00*** ed il valore di tempo nella variabile ***VarOut*** inizia il conteggio. Raggiunto il tempo definito 5 secondi, l'uscita ***Do00M00*** si azzera mentre il valore di tempo nella variabile ***VarOut*** rimane bloccato sul valore di preset (5000 mS) sino alla disattivazione dell'ingresso ***Di00M00***.

Disattivando l'ingresso digitale ***Di00M00*** durante la temporizzazione non si hanno ripercussioni nè sullo stato dell'uscita ***Do00M00***, nè sul valore della variabile ***VarOut***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBeTP	eTP	Auto	No	0	..	eTP (Timer pulse function block)
2	OutValue	UDINT	Auto	No	0	..	Output value

Esempio LD *(PTP115A100, eTP_LD)*

Esempio IL

```
CAL FBeTP (* Call the eTP function block *)

LD  Di00M00
ST  FBeTP.IN (* Transfer the digital input to timer input *)

LD  5000
ST  FBeTP.PT (* Set the delay time *)

LD  FBeTP.Q
ST  Do00M00 (* When time is passed the digital output is set *)

LD  FBeTP.ET
ST  OutValue (* The counting time is copied to variable *)
```

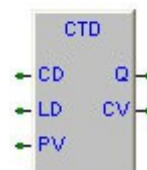
Esempio ST

```
FBeTP(PT:=5000); (* Call the eTP function block *)

FBeTP.IN:=Di00M00; (* Transfer the digital input to timer input *)
OutValue:=FBeTP.ET; (* The counting time is copied to variable *)
Do00M00:=FBeTP.Q; (* When time is passed the digital output is set *)
```

7.3 Funzioni ed FB per gestione counters

7.3.1 CTD, Counter Down

Type	Library
FB	ePLCStdLib_B000

Questo blocco funzione esegue la gestione di un counter in decremento. Agendo sull'ingresso di load ***LD*** è possibile in qualsiasi momento trasferire il valore di preset definito su ***PV*** nel counter ***CV***. Ad ogni fronte di attivazione dell'ingresso ***CD***, il valore del counter ***CV*** viene decrementato, quando il valore raggiunge 0, l'uscita ***Q*** viene settata ed il conteggio si arresta. Solo agendo sull'ingresso di load ***LD*** è possibile presettare il counter e fare ripartire un nuovo conteggio.

L'ingresso di load LD è prioritario sull'ingresso di decremento CD.

CD (BOOL) Comando decremento counter, ad ogni fronte attivazione il valore del counter CV si decrementa.

LD (BOOL) Comando di load, attivando l'ingresso il valore di preset PV, viene trasferito nel valore del counter CV.

PV (INT) Valore di preset, attivando l'ingresso di load LD, viene trasferito nel valore del counter CV.

Q (BOOL) Uscita counter, attiva se il valore del counter CV raggiunge il valore 0.

CV (INT) Valore counter, valore di conteggio counter, quando raggiunge il valore 0, l'uscita Q si attiva ed il counter non si decrementa più.

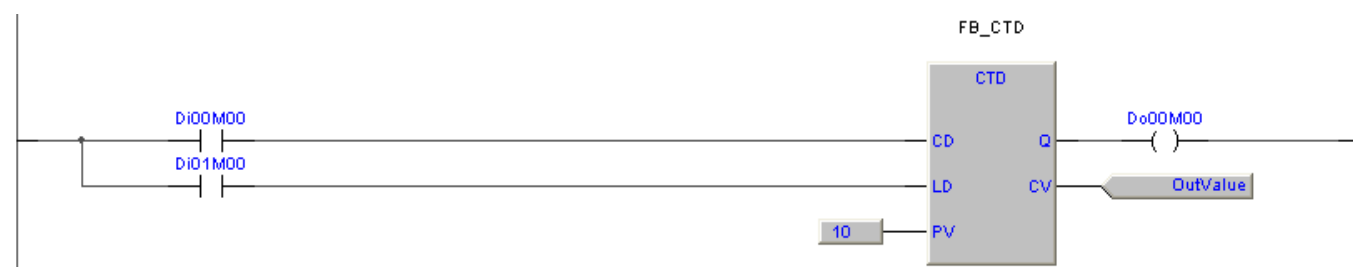
Esempi

Il counter è presettato a 10 ed il suo valore in uscita CV è copiato nella variabile ***OutValue***. Attivando l'ingresso digitale ***Di01M00*** il counter viene presettato ed il suo valore in uscita CV è posto a 10, resettando anche l'uscita Q.

Sul fronte di attivazione dell'ingresso digitale ***Di00M00*** il counter è decrementato di 1, quando il valore di conteggio si azzera, il conteggio si arresta e viene attivata l'uscita del counter Q che attiva l'uscita digitale ***Do00M00***. Per fare ripartire il conteggio occorre attivare l'ingresso digitale ***Di01M00*** che presetta il counter.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FB_CTD	CTD	Auto	No	0	..	CTD (Counter Down function block)
2	OutValue	INT	Auto	No	0	..	Output value

Esempio LD *(PTP115A100, CTD_LD)*

Esempio IL

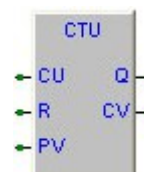
```
CAL FB_CTD (* Call the CTD function block *)
LD 10
ST FB_CTD.PV (* Preset value *)

LD Di00M01
ST FB_CTD.CD (* On the raising edge of digital input the counter count down *)

LD Di01M00
ST FB_CTD.LD (* If the digital input is set the PV value is loaded *)

LD FB_CTD.Q
ST Do00M00 (* If the counter value is 0 the digital output is set *)

LD FB_CTD.CV
ST OutValue (* The counter value is copied to the variable *)
```

Esempio ST

```
FB_CTD(PV:=10); (* Call the CTD function block and preset counter *)

FB_CTD.CD:=Di00M00; (* On the raising edge of digital input the counter count down *)
FB_CTD.LD:=Di01M00; (* If the digital input is set the PV value is loaded *)
Do00M00:=FB_CTD.Q; (* If the counter value is 0 the digital output is set *)
OutValue:=FB_CTD.CV; (* The counter value is copied to the variable *)
```

7.3.2 CTU, Counter Up

Type	Library
FB	ePLCStdLib_B000

Questo blocco funzione esegue la gestione di un counter in incremento. Agendo sull'ingresso di reset ***R*** è possibile in qualsiasi momento azzerare il valore del counter ***CV***. Ad ogni fronte di attivazione dell'ingresso ***CU***, il valore del counter ***CV*** viene incrementato. Quando il valore del counter ***CV*** raggiunge il valore di preset, definito su ***PV***, l'uscita ***Q*** viene settata ed il conteggio si arresta. Solo agendo sull'ingresso di reset ***R*** si potrà resettare il counter e fare ripartire un nuovo conteggio.

L'ingresso di reset R è prioritario sull'ingresso di incremento CU.

CU (BOOL) Comando incremento counter, ad ogni fronte attivazione il valore del counter CV si incrementa.

R (BOOL) Comando di reset, attivando l'ingresso il valore del counter CV si resetta.

PV (INT) Valore di preset, quando il valore del counter CV raggiunge questo valore l'uscita Q si attiva ed il counter non si incrementa più.

Q (BOOL) Uscita counter, attiva se il valore del counter CV raggiunge il valore definito in preset PV.

CV (INT) Valore counter, valore di conteggio counter, quando raggiunge il valore di preset PV, l'uscita Q si attiva ed il counter non si incrementa più.

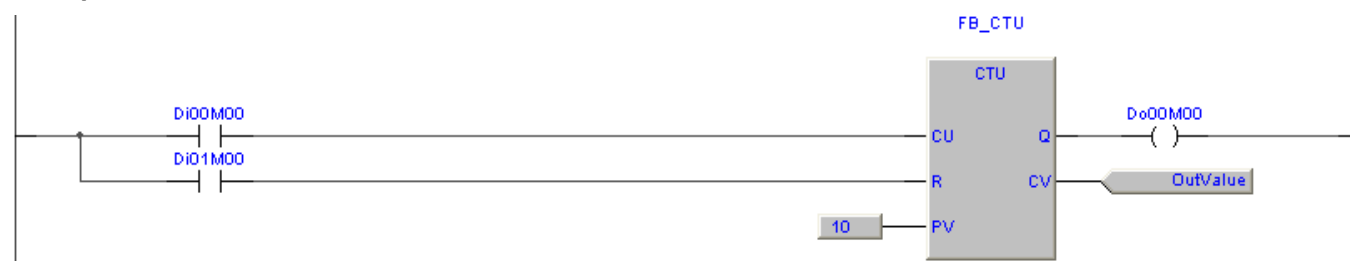
Esempi

Il counter è presettato a 10 ed il suo valore in uscita CV è copiato nella variabile ***OutValue***. Attivando l'ingresso digitale ***Di01M00*** il counter viene resettato ed il suo valore in uscita CV è posto a 0, resettando anche l'uscita Q.

Sul fronte di attivazione dell'ingresso digitale ***Di00M00*** il counter è incrementato di 1, quando il valore di conteggio raggiunge il valore di preset, il conteggio si arresta e viene attivata l'uscita del counter Q che attiva l'uscita digitale ***Do00M00***. Per fare ripartire il conteggio occorre attivare l'ingresso digitale ***Di01M00*** che resetta il counter.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FB_CTU	CTU	Auto	No	0	..	CTU (Counter Up function block)
2	OutValue	INT	Auto	No	0	..	Output value

Esempio LD *(PTP115A100, CTU_LD)*

Esempio IL

```
CAL FBCTU (* Call the CTU function block *)
LD  10
ST  FBCTU.PV (* Preset counter *)

LD Di00M00
ST FB_CTU.CU (* On the raising edge of digital input the counter count up *)

LD Di01M00
ST FB_CTU.R (* If the digital input is set the counter is reset *)

LD FBCTU.Q
ST Do00M00 (* If the counter value has reached the preset the digital output is set *)

LD FB_CTU.CV
ST OutValue (* The counter value is copied to the variable *)
```

7.3.3 CTUD, Counter Up/Down

Type	Library
FB	ePLCStdLib_B000

Questo blocco funzione esegue la gestione di un counter in incremento e decremento. Agendo sull'ingresso di reset R è possibile in qualsiasi momento azzerare il valore del counter ***CV***. Agendo sull'ingresso di load ***LD*** è possibile in qualsiasi momento trasferire il valore di preset definito su ***PV*** nel counter ***CV***.

Ad ogni fronte di attivazione dell'ingresso ***CU***, il valore del counter ***CV*** viene incrementato. Quando il valore del counter ***CV*** raggiunge il valore di preset, definito su ***PV***, l'uscita ***Q*** viene settata ed il conteggio si arresta. Solo agendo sull'ingresso di reset ***R*** si potrà resettare il counter e fare ripartire un nuovo conteggio.

Ad ogni fronte di attivazione dell'ingresso ***CD***, il valore del counter ***CV*** viene decrementato, quando il valore raggiunge 0, l'uscita ***Q*** viene settata ed il conteggio si arresta. Solo agendo sull'ingresso di load ***LD*** è possibile presettare il counter e fare ripartire un nuovo conteggio.

CU (BOOL) Comando incremento counter, ad ogni fronte attivazione il valore del counter CV si incrementa

CD (BOOL) Comando decremento counter, ad ogni fronte attivazione il valore del counter CV si decrementa.

R (BOOL) Comando di reset, attivando l'ingresso il valore del counter CV si resetta.

LD (BOOL) Comando di load, attivando l'ingresso il valore di preset PV, viene trasferito nel valore del counter CV.

PV (INT) Valore di preset, quando il valore del counter CV raggiunge questo valore l'uscita Q si attiva ed il counter non si incrementa più.

QU (BOOL) Uscita counter up, attiva se il valore del counter CV raggiunge il valore definito in preset PV.

QD (BOOL) Uscita counter down, attiva se il valore del counter CV raggiunge il valore 0.

CV (INT) Valore counter, valore di conteggio counter, quando raggiunge il valore di preset PV, l'uscita Q si attiva ed il counter non si incrementa più.

Esempi

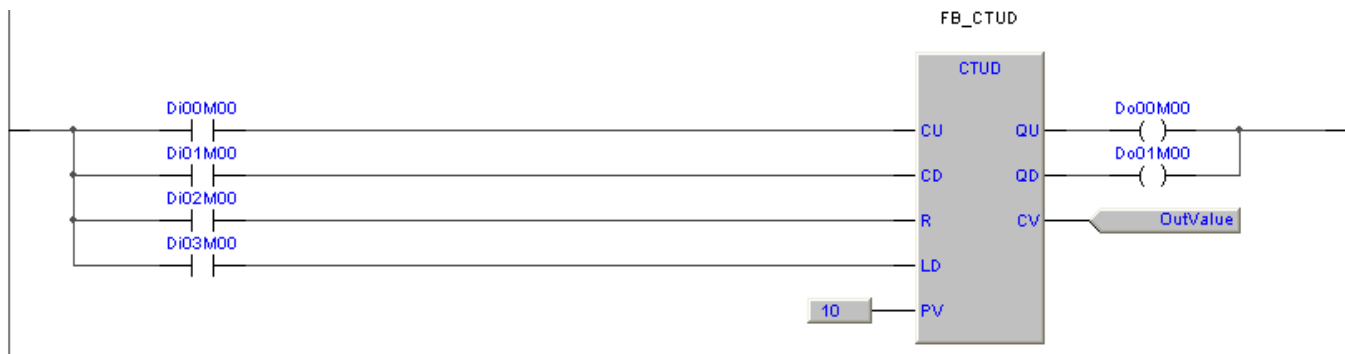
Il counter è presettato a 10 ed il suo valore in uscita CV è copiato nella variabile ***VarOut***. Attivando l'ingresso digitale ***Di02M00*** il counter viene resettato ed il suo valore in uscita CV è posto a 0, resettando anche l'uscita QU. Attivando l'ingresso digitale ***Di03M00*** il counter viene presettato ed il suo valore in uscita CV è posto a 10, resettando anche l'uscita QD.

Sul fronte di attivazione dell'ingresso digitale ***Di00M00*** il counter è incrementato di 1, quando il valore di conteggio raggiunge il valore di preset, il conteggio si arresta e viene attivata l'uscita del counter QU che attiva l'uscita digitale ***Do00M00***. Per fare ripartire il conteggio occorre attivare l'ingresso digitale ***Di02M00*** che resetta il counter.

Sul fronte di attivazione dell'ingresso digitale ***Di01M00*** il counter è decrementato di 1, quando il valore di conteggio si azzera, il conteggio si arresta e viene attivata l'uscita del counter QD che attiva l'uscita digitale ***Do01M00***. Per fare ripartire il conteggio occorre attivare l'ingresso digitale ***Di03M00*** che presetta il counter.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FB_CTUD	CTUD	Auto	No	0	..	CTUD (Counter Up/Down function block)
2	OutValue	INT	Auto	No	0	..	Output value

Esempio LD *(PTP115A100, CTUD_LD)*

Esempio IL

```
CAL FB_CTUD (* Call the CTUD function block *)
LD 10
ST FB_CTUD.PV (* Preset value *)

LD Di00M00
ST FB_CTUD.CU (* On the raising edge of digital input the counter count up *)

LD Di01M00
ST FB_CTUD.CD (* On the raising edge of digital input the counter count down *)

LD Di02M00
ST FB_CTUD.R (* If the digital input is set the counter is reset *)

LD Di03M00
ST FB_CTUD.LD (* If the digital input is set the PV value is loaded *)

LD FB_CTUD.QU
ST Do00M00 (* If the counter value has reached the preset the digital output is set *)

LD FB_CTUD.QD
ST Do01M00 (* If the counter value is 0 the digital output is set *)

LD FB_CTUD.CV
ST OutValue (* The counter value is copied to the variable *)
```

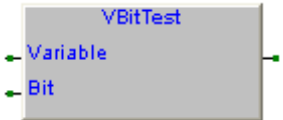
Esempio ST

```
FB_CTUD(PV:=10); (* Call the CTD function block and preset counter *)

FB_CTUD.CU:=Di00M00; (* On the raising edge of digital input the counter count up *)
FB_CTUD.CD:=Di01M00; (* On the raising edge of digital input the counter count down *)
FB_CTUD.R:=Di02M00; (* If the digital input is set the counter is reset *)
FB_CTUD.LD:=Di03M00; (* If the digital input is set the PV value is loaded *)
Do00M00:=FB_CTUD.QU; (* If the counter value has reached the preset the digital output is set *)
Do01M00:=FB_CTUD.QD; (* If the counter value is 0 the digital output is set *)
OutValue:=FBCTUD.CV; (* The counter value is copied to the variable *)
```

7.4 Funzioni ed FB per conversione dati

7.4.1 VBitTest, Variable bit test

Type	Library
Function	ePLCUtyLib_C000

Questa funzione esegue il test di un bit in una variabile.

Parametri funzione:

Variable (UDINT) Variabile in cui testare il bit.

Bit (USINT) Numero del bit da testare (Range da 0 a 31).

La funzione ritorna:

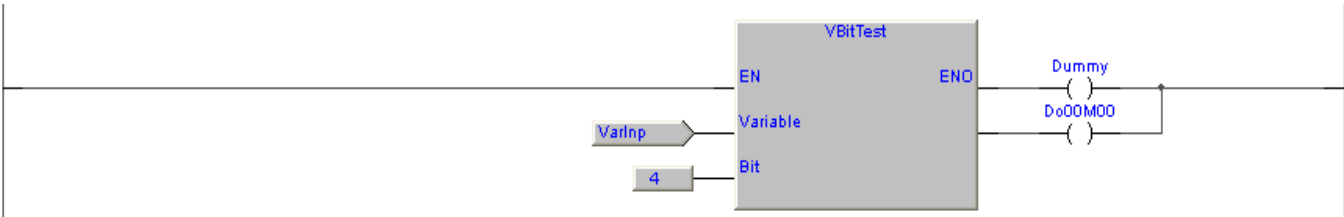
(BOOL) Stato bit indicato.

Esempi

Lo stato del bit 4 della variabile ***VarInp*** viene trasferito sull'uscita digitale ***Do00M00***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
2	VarInp	UDINT	Auto	No	0	..	Variable input

Esempio LD

Esempio IL

```
LD VarInp (* Variable input *)
VBitTest 4 (* Variable bit test *)
ST Do00M00 (* Transfer bit status to digital output *)
```

Esempio ST

```
Do00M00:=VBitTest(VarInp, 4); (* Variable bit test *)
```

Type	Library
Function	ePLCUtyLib_C000

7.4.2 VBitSet, Variable bit set

Questa funzione esegue il set di un bit in una variabile.

Parametri funzione:

Value (BOOL) Valore bit da settare.

Variable (UDINT) Variabile in cui settare il bit.

Bit (USINT) Numero del bit da settare (Range da 0 a 31).

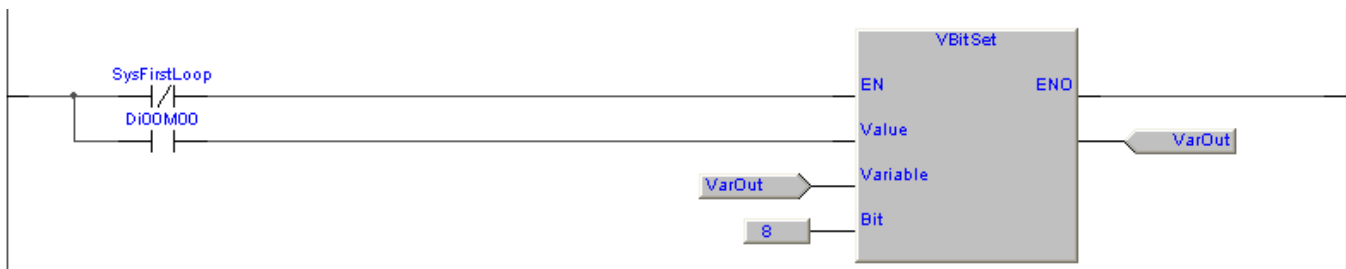
La funzione ritorna:

(UDINT) Valore variabile dopo il set del bit.

Esempi

Lo stato dell'ingresso digitale ***Di00M00*** è trasferito nel bit 8 della variabile ***VarOut***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarOut	UDINT	Auto	No	0	..	Variable output

Esempio LD

Esempio IL

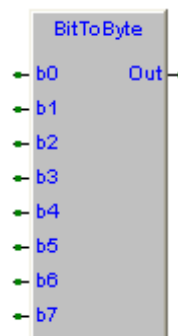
```
LD Di00M00 (* Variable input *)
VBitSet VarOut, 8 (* Variable bit set *)
ST VarOut (* Transfer result to variable *)
```

Esempio ST

```
VarOut:=VBitSet(Di00M00, VarOut, 8); (* Variable bit set *)
```

7.4.3 BitToByte, Bit to byte conversion

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione permette di convertire 8 variabili ***BOOL*** in una variabile ***BYTE***.

b0 (BOOL) Bit 0 del byte di ***Out***.

... ...

b7 (BOOL) Bit 7 del byte di ***Out***.

Out (BYTE) Risultato conversione ingressi a bit.

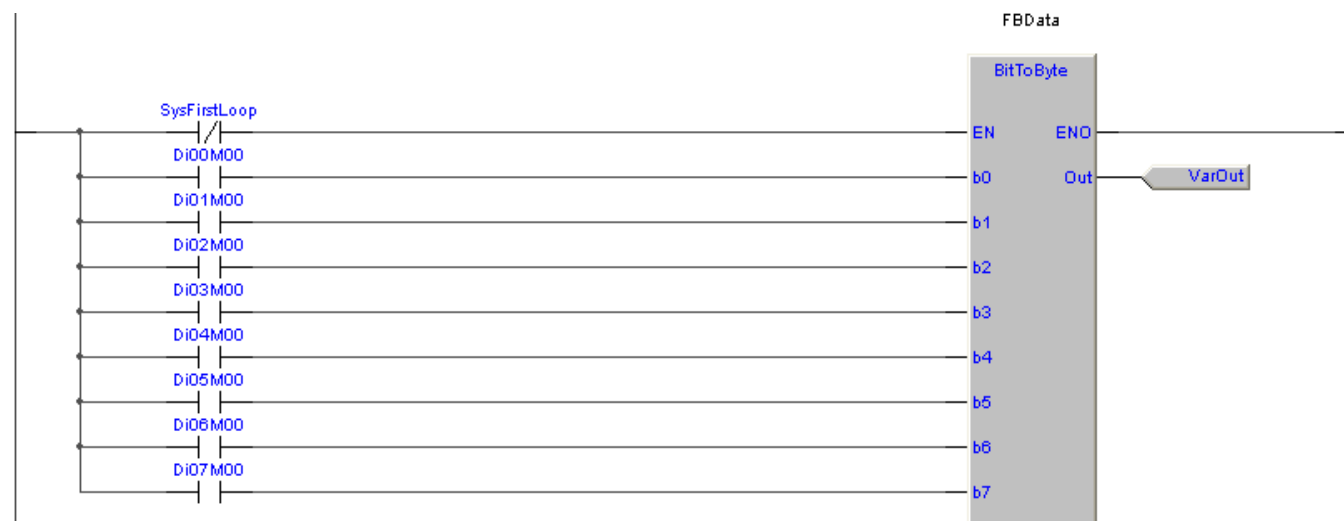
Esempi

Gli 8 ingressi digitali del modulo 0 sono trasferiti nella variabile ***VarOut***. Attivando il solo ingresso digitale ***Di00M00*** la variabile ***VarOut*** assumerà valore 1, attivando il solo ingresso digitale ***Di01M00*** la variabile ***VarOut*** assumerà valore 2, e così via fino all'ingresso ***Di07M00*** attivando il quale la variabile ***VarOut*** assumerà valore 128. Attivando più ingressi contemporaneamente la variabile ***VarOut*** assumerà valore pari alla somma degli ingressi attivati.

Per semplicità negli esempi IL e ST non vengono riportati tutti i bit.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	BitToByte	Auto	No	0	..	FB Bit to byte data
2	VarOut	UDINT	Auto	No	0	..	Variable output

Esempio LD *(PTP114A100, LD_BitToByte)*

Esempio IL *(PTP114A100, IL_BitToByte)*

```
LD  Di00M00
ST  FBData.b0 (* Transfer digital input to input bit *)

LD  Di07M00
ST  FBData.b7 (* Transfer digital input to input bit *)

CAL FBData (* Call the BitToByte function block *)

LD  FBData.Out
ST  VarOut (* Transfer the result to variable *)
```

Esempio ST *(PTP114A100, ST_BitToByte)*

```
FBData.b0:=Di00M00; (* Transfer digital input to input bit *)
FBData.b7:=Di07M00; (* Transfer digital input to input bit *)

FBData(); (* Call the BitToByte function block *)

VarOut:=FBData.Out; (* Transfer the result to variable *)
```

7.4.4 ByteToBit, Byte to bit conversion

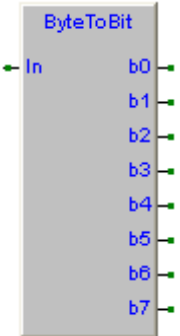
Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione permette di convertire una variabile ***BYTE*** in 8 variabili ***BOOL***.

In (BYTE) Valore byte da convertire

b0 (BOOL) Bit 0 di ***In***.

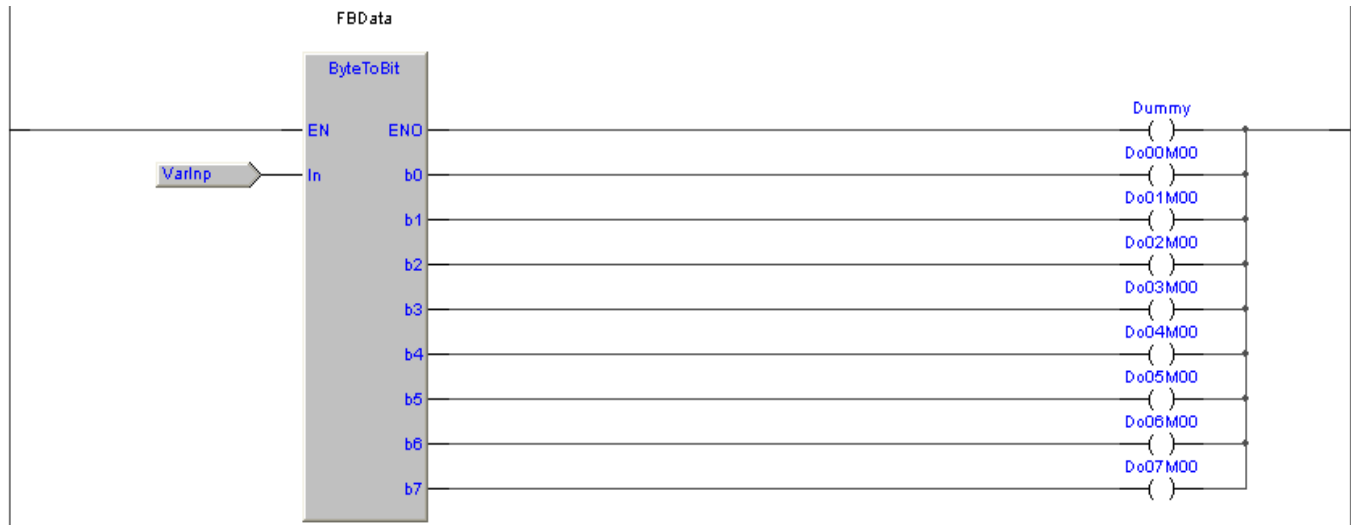
... ...

b7 (BOOL) Bit 7 di ***In***.

Esempi

Lo stato del bit 0 della variabile ***VarInp*** viene trasferito sull'uscita digitale ***Do00M00*** lo stato del bit 1 della variabile ***VarInp*** viene trasferito sull'uscita digitale ***Do01M00*** e così via fino allo stato del bit 7 della variabile ***VarInp*** viene trasferito sull'uscita digitale ***Do07M00***. Per semplicità negli esempi IL e ST non vengono riportati tutti i bit.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarInp	USINT	Auto	No	0	..	Variable input
2	FBData	ByteToBit	Auto	No	0	..	FB Byte to bit data
3	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable

Esempio LD *(PTP114A100, LD_ByteToBit)*

Esempio IL *(PTP114A100, IL_ByteToBit)*

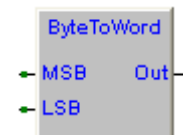
```
LD  VarInp
ST  FBData.In (* Transfer the variable to input *)

CAL FBData (* Call the ByteToBit function block *)

LD  FBData.b0
ST  Di00M00 (* Transfer output bit to digital output *)
```

Esempio ST *(PTP114A100, ST_ByteToBit)*

```
FBData(In:=VarInp); (* Call the ByteToBit function block *)

Do00M00:=FBData.b0; (* Transfer output bit to digital output *)
Do01M00:=FBData.b1; (* Transfer output bit to digital output *)
```

7.4.5 ByteToWord, Byte to word conversion

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione permette di convertire due variabili ***BYTE*** in una variabile ***WORD***.

MSB (BYTE) MSB del valore in uscita ***Out***

LSB (BYTE) LSB del valore in uscita ***Out***

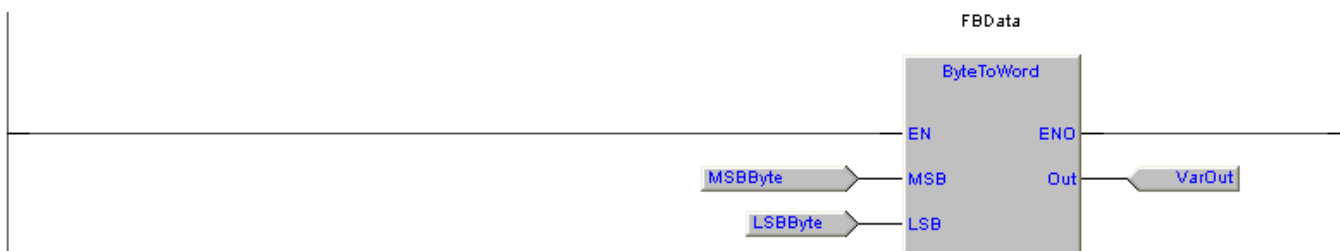
Out (WORD) Valore in uscita

Esempi

Le due variabili ***MSBByte*** e ***LSBByte*** sono uniti nella variabile ***VarOut*** in uscita.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	ByteToWord	Auto	No	0	..	FB Byte to word data
2	LSBByte	BYTE	Auto	No	0	..	Valore byte MSB
3	MSBByte	BYTE	Auto	No	0	..	Valore byte LSB
4	VarOut	WORD	Auto	No	0	..	Variable output

Esempio LD *(PTP114A100, LD_ByteToWord)*

Esempio IL *(PTP114A100, IL_ByteToWord)*

```
LD   MSBByte
ST   FBData.MSB (* Transfer the MSB variable to input *)

LD   LSBByte
ST   FBData.LSB (* Transfer the LSB variable to input *)

CAL FBData  (* Call the ByteToWord function block *)

LD   FBData.Out
ST   VarOut  (* Transfer output to variable *)
```

Esempio ST *(PTP114A100, ST_ByteToWord)*

```
FBData.MSB:=MSBByte; (* Transfer the MSB variable to input *)
FBData.LSB:=LSBByte; (* Transfer the LSB variable to input *)

FBData(); (* Call the ByteToWord function block *)

VarOut:=FBData.Out; (* Transfer output to variable *)
```

Type	Library
FB	ePLCUtyLib_C000

7.4.6 WordToByte, Word to byte conversion

Questo blocco funzione permette di convertire una variabile ***WORD*** in due variabili ***BYTE***.

IN (WORD) Variabile da convertire.

MSB (BYTE) MSB del valore in ingresso.

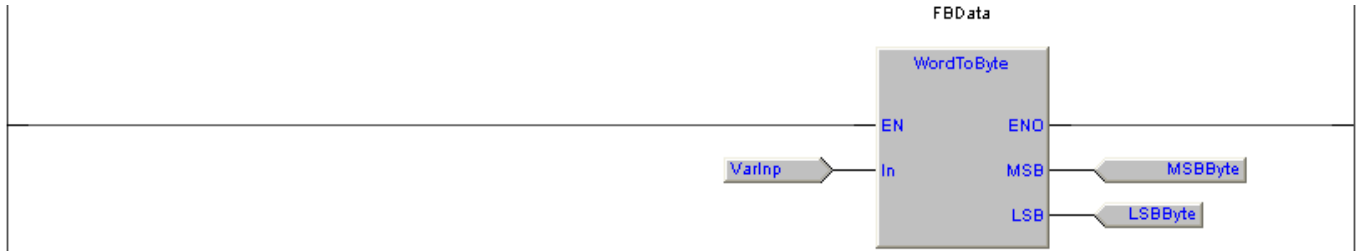
LSB (BYTE) LSB del valore in ingresso.

Esempi

Le variabile ***VarInp*** è divisa nelle due variabili ***MSBByte*** e ***LSBByte***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	LSBByte	BYTE	Auto	No	0	..	Valore byte MSB
2	MSBByte	BYTE	Auto	No	0	..	Valore byte LSB
3	VarInp	WORD	Auto	No	0	..	Variable input
4	FBData	WordToByte	Auto	No	0	..	FB Word to byte data

Esempio LD *(PTP114A100, LD_WordToByte)*

Esempio IL *(PTP114A100, IL_WordToByte)*

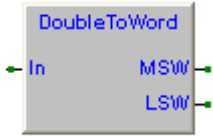
```
LD  VarInp
ST  FBData.In (* Transfer the variable to input *)

CAL FBData (* Call the WordToByte function block *)

LD  FBData.MSB
ST  MSBByte (* Transfer the MSB output to variable *)

LD  FBData.LSB
ST  LSBByte (* Transfer the LSB output to variable *)
```

Esempio ST *(PTP114A100, ST_WordToByte)*

```
FBData.In:=VarInp; (* Transfer the variable to input *)

FBData(); (* Call the WordToByte function block *)

MSBByte:=FBData.MSB; (* Transfer the MSB output to variable *)
LSBByte:=FBData.LSB; (* Transfer the LSB output to variable *)
```

7.4.7 DoubleToWord, Double to word conversion

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione permette di convertire una variabile DWORD in due variabili WORD.

IN (DWORD) Variabile da convertire.

MSW (WORD) MSW del valore in ingresso.

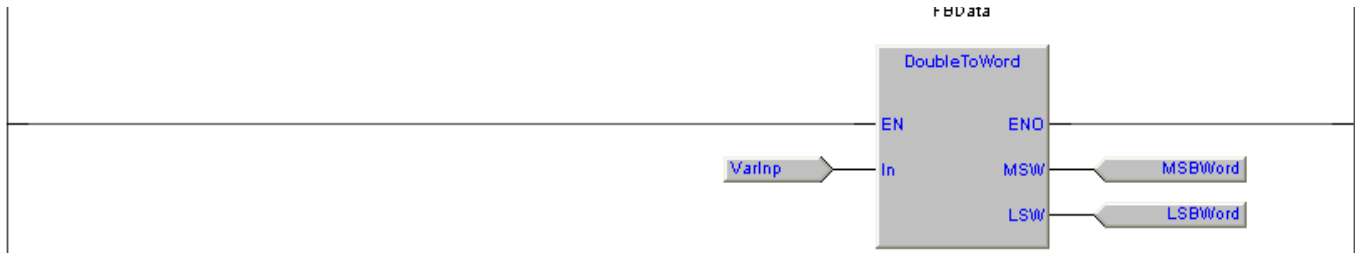
LSW (WORD) LSW del valore in ingresso.

Esempi

Le variabile ***VarInp*** è divisa nelle due variabili ***MSBWord*** e ***LSBWord***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	LSBWord	WORD	Auto	No	0	..	Valore word MSB
2	MSBWord	WORD	Auto	No	0	..	Valore word LSB
3	VarInp	DWORD	Auto	No	0	..	Variable input
4	FBData	DoubleToWor	Auto	No	0	..	FB Word to byte data

Esempio LD *(PTP114A100, LD_DoubleToWord)*

Esempio IL *(PTP114A100, IL_DoubleToWord)*

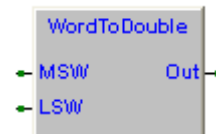
```
LD  VarInp
ST  FBData.In (* Transfer the variable to input *)

CAL FBData (* Call the DoubleToWord function block *)

LD  FBData.MSW
ST  MSBWord (* Transfer the MSW output to variable *)

LD  FBData.LSW
ST  LSBWord (* Transfer the LSW output to variable *)
```

Esempio ST *(PTP114A100, ST_DoubleToWord)*

```
FBData.In:=VarInp; (* Transfer the variable to input *)

FBData(); (* Call the DoubleToWord function block *)

MSBWord:=FBData.MSW; (* Transfer the MSW output to variable *)
MSBWord:=FBData.LSW; (* Transfer the LSW output to variable *)
```

7.4.8 WordToDouble, Word to double conversion

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione permette di convertire due variabili ***WORD*** in una variabile ***DWORD***.

MSW (WORD) MSB del valore in uscita ***Out***

LSW (WORD) LSB del valore in uscita ***Out***

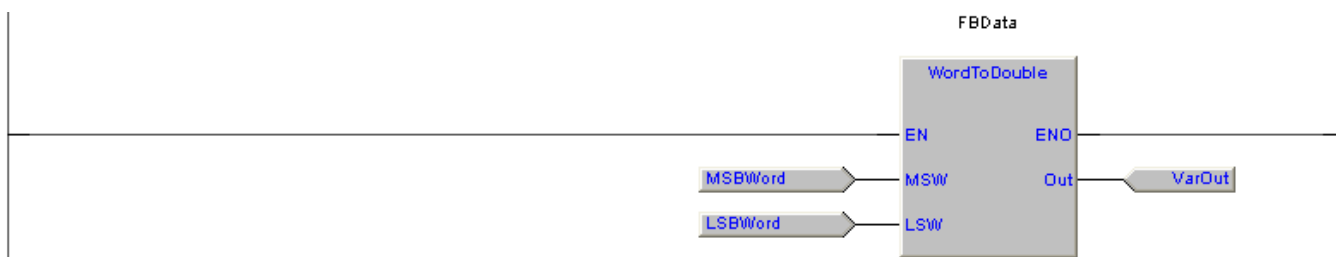
Out (DWORD) Valore in uscita

Esempi

Le due variabili ***MSBWord*** e ***LSBWord*** sono uniti nella variabile ***VarOut*** in uscita.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	LSBWord	WORD	Auto	No	0	..	Valore word MSB
2	MSBWord	WORD	Auto	No	0	..	Valore word LSB
3	VarOut	DWORD	Auto	No	0	..	Variable output
4	FBData	WordToDouble	Auto	No	0	..	FB Word to double data

Esempio LD *(PTP114A100, LD_WordToDouble)*

Esempio IL *(PTP114A100, IL_WordToDouble)*

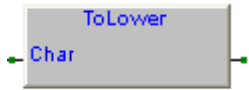
```
LD   MSBWord
ST   FBData.MSW (* Transfer the MSW variable to input *)

LD   LSBWord
ST   FBData.LSW (* Transfer the LSW variable to input *)

CAL FBData  (* Call the WordToDouble function block *)

LD   FBData.Out
ST   VarOut  (* Transfer output to variable *)
```

Esempio ST *(PTP114A100, ST_WordToDouble)*

```
FBData.MSW:=MSBWord; (* Transfer the MSW variable to input *)
FBData.LSW:=LSBWord; (* Transfer the LSW variable to input *)

FBData(); (* Call the WordToDouble function block *)

VarOut:=FBData.Out; (* Transfer output to variable *)
```

7.4.9 ToLower, Uppercase to lowercase letter conversion

Type	Library
Function	ePLCUtyLib_C000

Questa funzione converte un carattere dal formato maiuscolo nel corrispondente carattere in formato minuscolo.

Parametri funzione:

Char (USINT) Carattere da convertire.

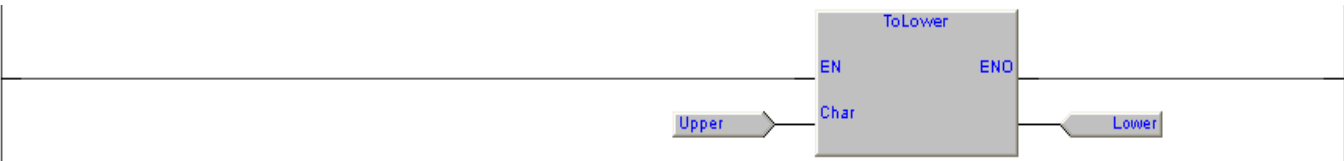
La funzione ritorna:

(USINT) Carattere nel formato minuscolo.

Esempi

La variabile ***Upper*** viene convertita nel corrispondente valore minuscolo e trasferita in ***Lower***. Il valore di inizializzazione 16#41 che corrisponde alla lettera ***A***, viene convertito nel valore 16#61 che corrisponde alla lettera ***a***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Upper	USINT	Auto	No	16#41	..	Uppercase letter
2	Lower	USINT	Auto	No	0	..	Lowercase letter

Esempio LD

Esempio IL

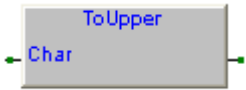
```
LD  Upper (* Uppercase letter *)
ToLower (* Uppercase to lowercase letter conversion *)
ST  Lower (* Lowercase letter *)
```

Esempio ST

```
Lower:=ToLower(Upper); (* Uppercase to lowercase letter conversion *)
```

7.4.10 ToUpper, Lowercase to uppercase letter conversion

Type	Library
Function	ePLCUtyLib_C000

Questa funzione converte un carattere dal formato minuscolo nel corrispondente carattere in formato maiuscolo.

Parametri funzione:

Char (USINT) Carattere da convertire.

La funzione ritorna:

(USINT) Carattere nel formato maiuscolo.

Esempi

La variabile ***Lower*** viene convertita nel corrispondente valore maiuscolo e trasferita in ***Upper***. Il valore di inizializzazione 16#61 che corrisponde alla lettera ***a***, viene convertito nel valore 16#41 che corrisponde alla lettera ***A***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Lower	USINT	Auto	No	16#61	..	Lowercase letter
2	Upper	USINT	Auto	No	0	..	Uppercase letter

Esempio LD

Esempio IL

```
LD  Lower (* Lowercase letter *)
ToUpper (* Lowercase to uppercase letter conversion *)
ST  Upper (* Lowercase letter *)
```

Esempio ST

```
Upper:=ToUpper(Lower); (* Lowercase to uppercase letter conversion *)
```

7.4.11 LEArrayToVar, Little endian array to variable conversion

Type	Library
Function	ePLCUtyLib_C000

Questa funzione acquisisce il valore da un array little endian (MSB - LSB) ***Source*** e lo trasferisce nella variabile di destinazione ***Destination*** in base al tipo ***Type*** di variabile definito.

LEArrayToVar
Type
Destination
Source

Parametri funzione:

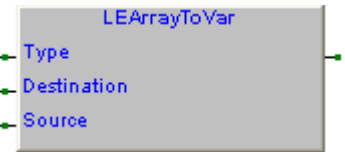
Type (USINT) Tipo di variabile destinazione come definito nella tabella **Variable types definition**.

Destination (@USINT) Indirizzo variabile di destinazione.

Source (@USINT) Indirizzo array sorgente.

La funzione ritorna:

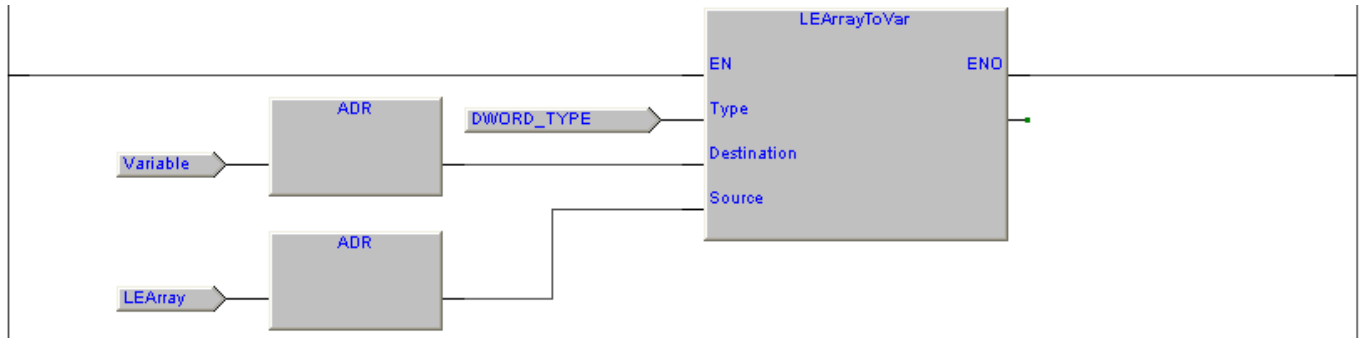
(BOOL) FALSE: Errore tipo dati, TRUE: Conversione eseguita.

Esempi

Il valore presente nell'array ***LEArray*** “1, 2, 3, 4” viene convertito nella variabile ***Variable***, eseguita la conversione la variabile avrà il valore 16#01020304.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Variable	DWORD	Auto	No	0	..	Variable
2	LEArray	USINT	Auto	[0..3]	1,2,3,4	..	Little endian array

Esempio LD

Type	Library
Function	ePLCUtyLib_C000

7.4.12 BEArrayToVar, Big endian array to variable conversion

Questa funzione acquisisce il valore da un array big endian (LSB - MSB) ***Source*** e lo trasferisce nella variabile di destinazione ***Destination*** in base al tipo ***Type*** di variabile definito.

Parametri funzione:

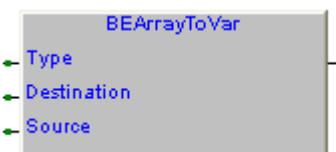
Type (USINT) Tipo di variabile destinazione come definito nella tabella **Variable types definition**.

Destination (@USINT) Indirizzo variabile di destinazione.

Source (@USINT) Indirizzo array sorgente.

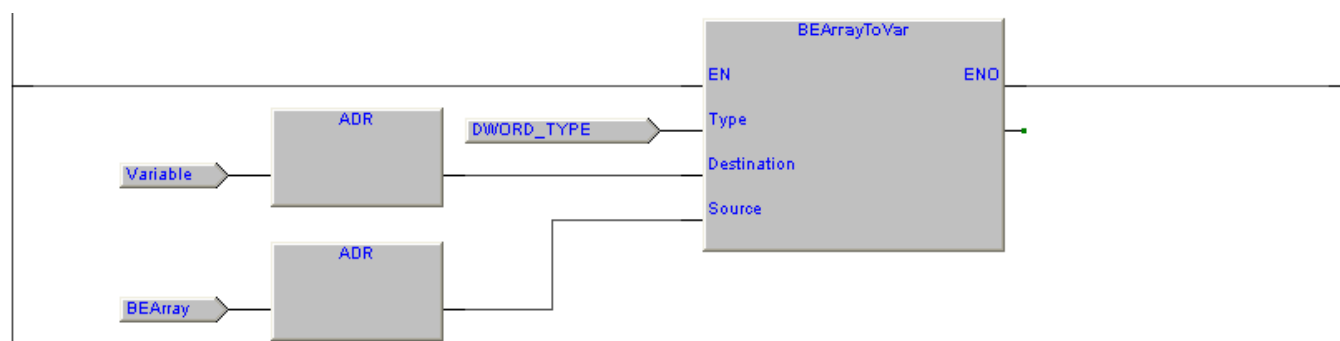
La funzione ritorna:

(BOOL) FALSE: Errore tipo dati, TRUE: Conversione eseguita.

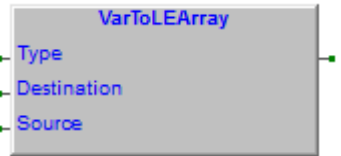
Esempi

Il valore presente nell'array ***BEArray*** “4, 3, 2, 1” viene convertito nella variabile ***Variable***, eseguita la conversione la variabile avrà il valore 16#01020304.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Variable	DWORD	Auto	No	0	..	Variable
2	BEArray	USINT	Auto	[0..3]	4,3,2,1	..	Big endian array

Esempio LD

7.4.13 VarToLEArray, variable to little endian array conversion

Type	Library
Function	ePLCUtyLib_C000

Questa funzione trasferisce il valore di una variabile ***Source*** in base al tipo ***Type*** definito, in un array ***Destination*** utilizzando il formato little endian (MSB - LSB).

Parametri funzione:

Type (USINT) Tipo di variabile come definito nella tabella **Variable types definition**.

Destination (@USINT) Indirizzo array di destinazione.

Source (@USINT) Indirizzo variabile sorgente.

La funzione ritorna:

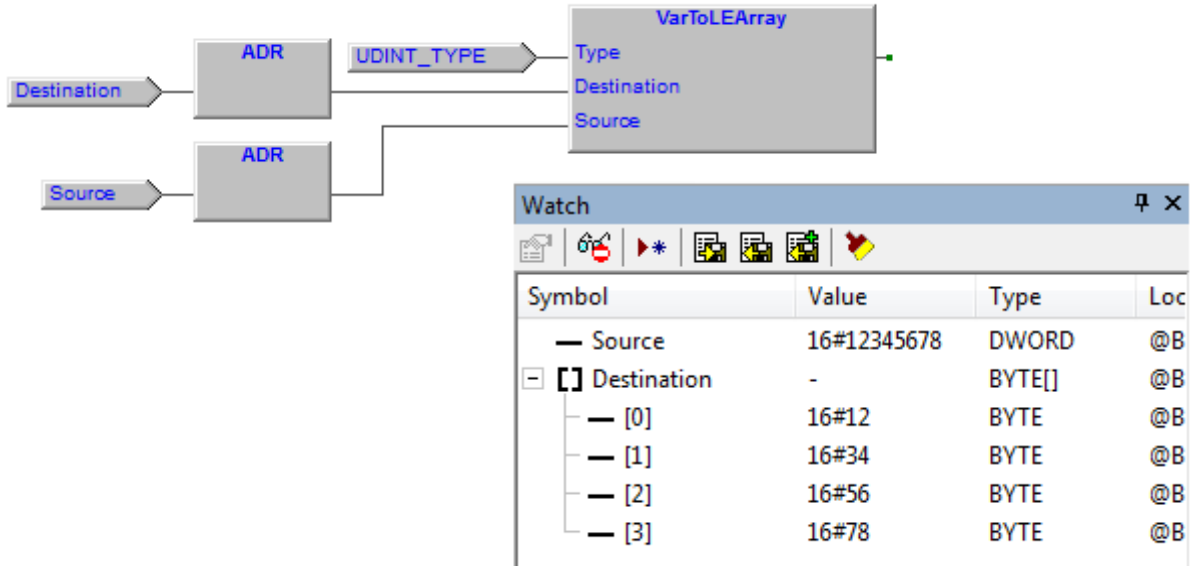
(BOOL) FALSE: Errore tipo dati, TRUE: Conversione eseguita.

Esempi

Il valore presente nella variabile ***Source*** viene convertito nell'array ***Destination*** di tipo little endian (MSB – LSB). Nell'esempio è riportata anche la finestra di Watch con i valori visualizzati.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Source	DWORD	Auto	No	16#12345678	..	Source variable
2	Destination	BYTE	Auto	[0..3]		..	Destination array

Esempio FBD

7.4.14 VarToBEArray, variable to big endian array conversion

Type	Library
Function	ePLCUtyLib_C000

Questa funzione trasferisce il valore di una variabile ***Source*** in base al tipo ***Type*** definito, in un array ***Destination*** utilizzando il formato big endian (LSB - MSB).

VarToBEArray
Type
Destination
Source

Parametri funzione:

Type (USINT) Tipo di variabile come definito nella tabella **Variable types definition**.

Destination (@USINT) Indirizzo array di destinazione.

Source (@USINT) Indirizzo variabile sorgente.

La funzione ritorna:

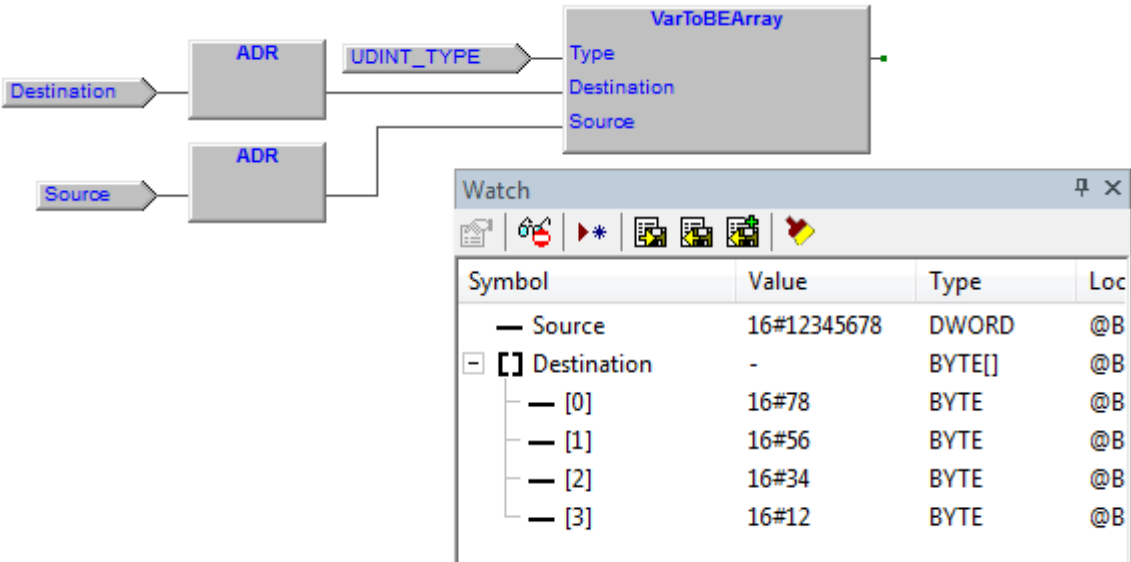
(BOOL) FALSE: Errore tipo dati, TRUE: Conversione eseguita.

Esempi

Il valore presente nella variabile ***Source*** viene convertito nell'array ***Destination*** di tipo big endian (LSB – MSB). Nell'esempio è riportata anche la finestra di Watch con i valori visualizzati.

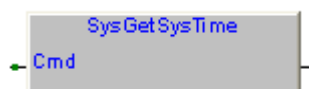
Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Source	DWORD	Auto	No	16#12345678	..	Source variable
2	Destination	BYTE	Auto	[0..3]		..	Destination array

Esempio FBD

7.5 Funzioni ed FB di utilità sistema

7.5.1 SysGetSysTime, get system time

Type	Library
Function	XTarget_07_0

Questa funzione ritorna il tempo di sistema espresso in µS. E' possibile definire con il valore di ***Cmd*** se si vuole avere il tempo di sistema attuale (***Cmd:=TRUE***) oppure quello memorizzato con la precedente esecuzione della funzione (***Cmd:=FALSE***).

Parametri funzione:

Cmd (BOOL) Indica il valore di tempo che deve essere ritornato.
TRUE: Viene salvato e ritornato il valore attuale di tempo.
FALSE: Viene ritornato il tempo salvato dalla precedente chiamata con ***Cmd:=TRUE***.

La funzione ritorna:

(UDINT) Tempo di sistema espresso in µS.

Esempi

Viene calcolato il tempo in cui l'ingresso digitale ***Di00M00*** rimane nella condizione di attivo.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	StartTime	UDINT	Auto	No	0	..	Input raising time reference (uS)
2	SetTime	UDINT	Auto	No	0	..	Input set time (uS)
3	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag

Esempio ST *(PTP116A000, ST_SysGetSysTime)*

```
(* Check if input is activated. *)

IF (Di00M00 <> Pulse) THEN
    Pulse:=Di00M00; (* Pulse flag *)

    (* On input raising edge relate time is saved. *)

    IF (Di00M00) THEN StartTime:=SysGetSysTime(TRUE); END_IF;

    (* On input falling edge the set time is calculated. *)

    IF (NOT(Di00M00)) THEN SetTime:=SysGetSysTime(TRUE)-StartTime; END_IF;

END_IF;
```

Calcolo timeout

Essendo il valore di tempo di sistema ritornato dalla funzione un numero ***UDINT*** che si incrementa ogni µS, ed al valore massimo esegue overflow a zero, non è possibile effettuare comparazioni dirette con il tempo di riferimento ma occorre sempre eseguire la differenza.

Nel seguente esempio viene attivato ***Timeout*** se l'ingresso ***Di00M00*** rimane attivo per più di un secondo.

```
IF NOT(Di00M00) THEN TimeBf:=SysGetSysTime(TRUE);
ELSE IF ((SysGetSysTime(TRUE)-TimeBf) >= 1000000) THEN Timeout:=TRUE; END_IF;
END_IF;
```

Lo stesso esempio scritto in questo modo non funziona correttamente.

```
IF NOT(Di00M00) THEN TimeBf:=SysGetSysTime(TRUE);
ELSE IF (SysGetSysTime(TRUE) >= (TimeBf+1000000)) THEN Timeout:=TRUE; END_IF;
END_IF;
```

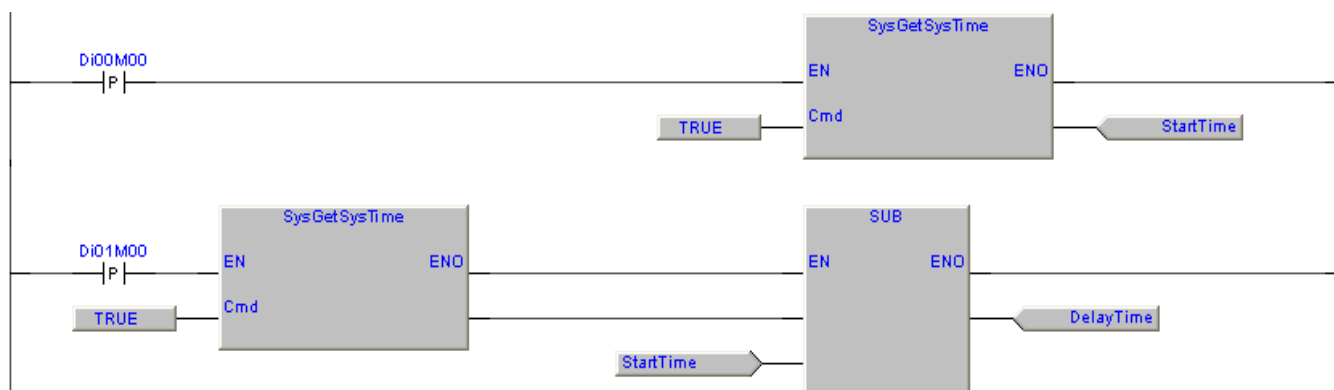
Semplice cronometro

Questo esempio realizza un semplice cronometro per misurare il tempo che intercorre tra un comando di start ed il comando di stop. Utilizzando ad esempio due fotocellule una sulla linea di start ed una sulla linea di stop di un percorso è possibile calcolare il tempo di percorrenza espresso in µS.

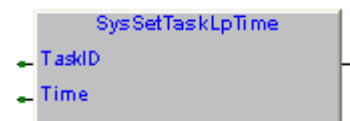
Attivando l'ingresso di start ***Di00M00*** viene salvato il tempo di sistema allo start nella variabile ***StartTime***, attivando l'ingresso di stop ***Di01M00*** viene calcolato il tempo trascorso tra il tempo salvato allo start ed il tempo nel momento di stop. Il tempo calcolato è salvato nella variabile ***DelayTime***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	StartTime	UDINT	Auto	No	0	..	Start time (uS)
2	DelayTime	UDINT	Auto	No	0	..	Delay time (uS)

Esempio LD *(PTP119A000)*

7.5.2 SysSetTaskLpTime, set task loop time

Type	Library
Function	XTarget_07_0

Questa funzione permette di impostare il tempo di esecuzione delle tasks PLC. Esistono due tasks eseguite a tempo la task slow ***ID_TASK_SLOW*** e la task fast ***ID_TASK_FAST***, ad ognuna di queste task può essere assegnato un tempo di esecuzione.

Se il tempo impostato non è compreso nel range definito o se il rapporto tra i tempi di esecuzione della task fast rispetto alla slow non sono coerenti la funzione non modifica i tempi di esecuzione e ritorna ***FALSE***. Di seguito sono riportati i range di tempo definibili per le varie tasks.

ID_TASK_FAST Range da 100 µS a 10 mS

ID_TASK_SLOW Range da 1 a 100 mS

Parametri funzione:

TaskID (USINT) Identifica la task a cui si vuole definire il tempo di esecuzione secondo le definizioni in Task ID.

Time (UDINT) Indica il valore di tempo di esecuzione task espresso in µS.

La funzione ritorna:

(BOOL) ***TRUE:*** Se funzione eseguita correttamente
FALSE: In caso di errore esecuzione, esempio parametri errati.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9948990 Non implementata nel simulatore.

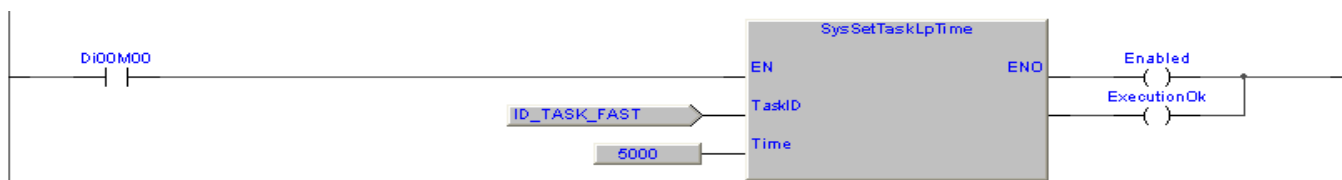
Esempi

Attivando l'ingresso ***Di00M00*** viene impostato un tempo di esecuzione di 5 ms per la task PlcFast.

Attenzione! Per aumentare i tempi di esecuzione delle tasks dal valore di default occorre eseguire la funzione nella task di boot.

Definizione variabili

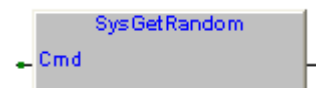
	Name	Type	Address	Array	Init value	Attribute	Description
1	Enabled	BOOL	Auto	No	FALSE	..	Function enabled
2	ExecutionOk	BOOL	Auto	No	FALSE	..	Function execution ok

Esempio LD *(PTP116A000, LD_SysSetTaskLpTime)*

Esempio ST

```
Enabled:=Di00M00; (* Function enabled *)

IF Di00M00 THEN
    ExecutionOk:=SysSetTaskLpTime(TaskID:=ID_TASK_FAST, Time:=5000); (* Function execution ok *)
END_IF;
```

7.5.3 SysGetRandom, get random number

Type	Library
Function	XTarget_07_0

Questa funzione ritorna un numero random compreso tra 0.0 e 1.0. E' possibile definire con il valore di ***Cmd*** se si vuole avere un nuovo numero random (***Cmd:=TRUE***) oppure quello memorizzato con la precedente esecuzione della funzione (***Cmd:=FALSE***).

Parametri funzione:

Cmd (BOOL) Indica il numero random ritornato.
TRUE: Viene salvato e ritornato un nuovo numero random.
FALSE: Viene ritornato il numero salvato dalla precedente chiamata con ***Cmd:=TRUE***.

La funzione ritorna:

(REAL) Un numero random compreso nel range da 0.0 a 1.0.

Esempi

Attivando l'ingresso digitale ***Di00M00*** viene inviato sulla porta seriale ***COM0*** una sequenza di 10 numeri random.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RandomNr	UINT	Auto	No	0	..	Random number
2	i	USINT	Auto	No	0	..	Auxiliary counter
3	NrOfChars	INT	Auto	No	0	..	Number of printed chars
4	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
5	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag

Esempio ST *(PTP116A000, ST_SysGetRandom)*

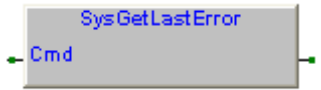
```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

    (* Check if input is activated. *)

    F (Di00M00 <> Pulse) THEN
        Pulse:=Di00M00; (* Pulse flag *)

    (* On input raising edge print out 10 random numbers. *)

        IF (Di00M00) THEN

            FOR i:=0 TO (9) BY 1 DO
                RandomNr:=TO_UINT(SysGetRandom(TRUE)*1000.0); (* Random number *)
                NrOfChars:=SysVarfprintf(Fp, 'Rn:%03d$r$n', UINT_TYPE, ADR(RandomNr));
            END_FOR;
        END_IF;
    END_IF;
```

Collegando un terminale seriale alla porta ***COM0*** impostato a ***115200,e,8,1*** vedremo un elenco del tipo:

```
Rn:437
Rn:488
Rn:898
...
Rn:261
Rn:944
```

7.5.4 SysGetLastError, get last error

Type	Library
Function	XTarget_07_0

Questa funzione ritorna il numero dell'ultimo errore rilevato da una funzione e/o da un blocco funzione. Occorre eseguire la funzione su abilitazione del bit di fault in uscita dalla funzione e/o blocco funzione da controllare. E' possibile definire con il valore di ***Cmd*** se si vuole avere il valore attuale dell'ultimo errore (***Cmd:=TRUE***) oppure quello memorizzato con la precedente esecuzione della funzione (***Cmd:=FALSE***).

Parametri funzione:

Cmd (BOOL) Indica il numero di errore ritornato.
TRUE: Viene ritornato l'ultimo valore di errore.
FALSE: Viene ritornato il numero salvato dalla precedente chiamata con ***Cmd:=TRUE***.

La funzione ritorna:

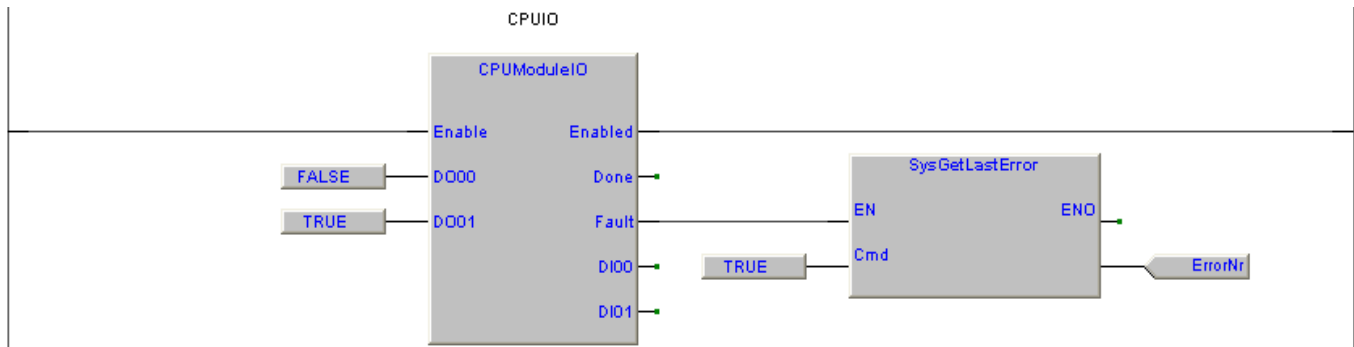
(UDINT) Il numero dell'ultimo errore rilevato

Esempi

Viene salvato l'eventuale errore durante l'esecuzione del blocco funzione ***CPUModuleIO***. In caso di errore il numero di errore è trasferito nella variabile ***ErrorNr***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	CPUIO	CPUModuleIO	Auto	No	0	..	I/O on CPU module
2	ErrorNr	UDINT	Auto	No	0	..	Error number

Esempio LD

7.5.5 SysPCodeAccept, accepts the protection code

Type	Library
Function	XTarget_07_0

Alcune funzioni di programma e/o blocchi funzione possono essere protetti da un codice che deve essere ordinato separatamente. Per abilitare l'esecuzione della funzione e/o del blocco funzione occorre sbloccarlo definendone il codice con questa funzione.

La funzione controlla il codice fornito e ritorna ***TRUE*** se codice accettato. Vedere capitolo ***Protezione funzioni e blocchi funzione*** per ulteriori informazioni.

Parametri funzione:

Code (STRING[20]) Codice di protezione.

La funzione ritorna:

(BOOL) ***TRUE:*** Codice verificato ID relativo sbloccato. ***FALSE:*** Codice non verificato.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9991100 Valore di ***Code*** non corretto.

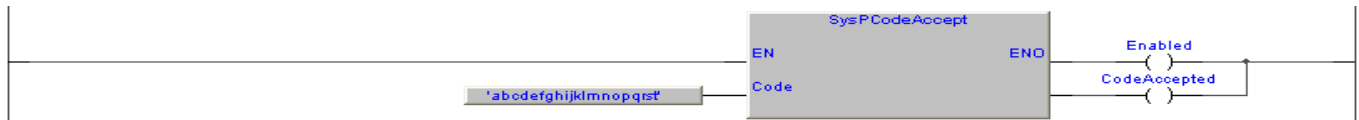
9991990 Non implementata nel simulatore.

Esempi

E' riportato un semplice programma che esegue il controllo sul codice di sblocco "***abcdefghijklmnopqrst***". Se il codice è corretto viene attivata la variabile ***CodeAccepted***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	CodeAccepted	BOOL	Auto	No	FALSE	..	Protection code accepted
2	Enabled	BOOL	Auto	No	FALSE	..	Function enabled

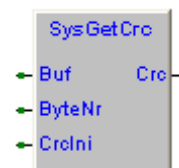
Esempio LD *(PTP116A000,LD_SysPCodeAccept)*

Esempio ST

```
(* Check the protection code. *)

    CodeAccepted:=SysPCodeAccept('abcdefghijklmnopqrst'); (* Protection code accepted *)
```

7.5.6 SysGetCrc, get CRC value

Type	Library
FB	XTarget_07_0

Questo blocco funzione esegue il calcolo del CRC ***Cyclic Redundancy Check***, (Controllo Ciclico di Ridondanza) su di un'area dati. Il calcolo è effettuato secondo le specifiche richieste dal protocollo ***modbus Rtu***

Occorre passare alla FB l'indirizzo del buffer di memoria ***Buf*** ed il numero di bytes ***ByteNr*** su cui eseguire il calcolo del CRC.

Buf (@USINT) Indirizzo dell'area di memoria su cui eseguire il calcolo del CRC.

ByteNr (UINT) Numero di bytes su cui eseguire il calcolo del CRC a partire dall'indirizzo definito in ***Buf***.

CRCIni (UINT) Valore di inizializzazione del CRC da calcolare.

CRC (UINT) Valore CRC calcolato.

Codici di errore

In caso di errore con ***SysGetLastError*** è possibile rilevare il codice di errore.

9978050 Errore allocazione blocco funzione.

9978070 Errore versione blocco funzione.

Esempi

Viene calcolato il CRC di un frame modbus Rtu per il comando di lettura registri ***Read holding registers***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Frame	USINT	Auto	[0..9]	10(0)	..	Frame array
2	RegsAddress	UINT	Auto	No	0	..	Registers address
3	NrOfRegs	USINT	Auto	No	0	..	Number of registers
4	GetCRC	SysGetCrc	Auto	No	0	..	Get CRC
5	CRCValue	UINT	Auto	No	0	..	CRC value

Esempio ST *(PTP116A100, ST_SysGetCrc)*

```
(* ---------------------------------------------------------------------------- *)
(* Calculate CRC of a modbus Rtu frame for command "Read holding registers".  *)
(* +--+--+--+--+--+--+--+ *)
(* |Nd|03|Addr |NumR |CRC| *)
(* +--+--+--+--+--+--+--+ *)
(* ---------------------------------------------------------------------------- *)
    (* Define the registers address and the number of registers to read. *)

    RegsAddress:=16#0120; (* Registers address *)
    NrOfRegs:=8; (* Number of registers *)

    (* Prepare the command frame. *)

    Frame[0]:=1; (* Node address *)
    Frame[1]:=3; (* Function code (16#03) *)
    Frame[2]:=TO_USINT(RegsAddress/256); (* MSB registers address *)
    Frame[3]:=TO_USINT(RegsAddress&255); (* LSB registers address *)
    Frame[4]:=0; (* MSB number of registers to read *)
    Frame[5]:=NrOfRegs; (* LSB number of registers to read *)

    (* Calculate the frame CRC. *)

    GetCRC.Buf:=ADR(Frame[0]); (* Buffer address *)
    GetCRC.ByteNr:=6; (* Byte number *)
    GetCRC.CrcIni:=16#FFFF; (* CRC ini value *)
    GetCRC(); (* Calculate CRC *)
    CRCValue:=GetCRC.Crc; (* CRC value *)
    Frame[6]:=TO_USINT(CRCValue/256); (* MSB of CRC value *)
    Frame[7]:=TO_USINT(CRCValue&255); (* LSB of CRC value *)
```

7.5.7 SysMAlloc, memory allocation

Type	Library
Function	XTarget_07_0

Questa funzione esegue l'allocazione di uno spazio di memoria della dimensione in byte definita da parametro ***Size***. La funzione ritorna il puntatore allo spazio di memoria allocato.

La memoria è allocata nella memoria di sistema e quindi non utilizza la memoria a disposizione del programma utente. Nel caso in cui non vi sia spazio in memoria per l'allocazione del buffer definito, la funzione ritorna ***0***.

Parametri funzione:

Size (UDINT) Dimensione in bytes dell'area da allocare.

La funzione ritorna:

(@USINT) Indirizzo allocazione buffer. ***NULL*** se non vi è spazio per allocare il buffer.

Codici di errore

In caso di errore la funzione torna ***NULL*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9947990 Non implementata nel simulatore.

Esempi

Su fronte attivazione ingresso ***Di00M00*** viene incrementata la variabile ***Counter*** e la stampa del suo valore trasferita nell'array ***StringOut***. Il valore presente in ***StringOut*** viene poi inviato sulla porta seriale ***COM0***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag
2	Ch	INT	Auto	No	0	..	Written character
3	i	INT	Auto	No	0	..	Auxiliary counter
4	NrOfChars	INT	Auto	No	0	..	Number of printed chars
5	Counter	UDINT	Auto	No	0	..	Counter
6	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
7	StringOut	@USINT	Auto	No	0	..	String output pointer

Esempio ST *(PTP116A300, ST_SysMAlloc)*

```
(* Here at first program execution loop allocate memory and open COM. *)

IF (SysFirstLoop) THEN
    StringOut:=SysMAlloc(16); (* String output pointer *)
    Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;

IF ((StringOut = 0) OR (Fp = 0)) THEN RETURN; END_IF;

(* On input raising edge the counter value is printed. *)

IF (Di00M00 <> Pulse) THEN
    Pulse:=Di00M00; (* Pulse flag *)

    IF (Di00M00) THEN
        Counter:=Counter+1; (* Counter *)
        NrOfChars:=SysVarsnprintf(StringOut, 32, 'Counter:%04d$r$n', UDINT_TYPE, ADR(Counter));

        FOR i:=0 TO NrOfChars DO
            Ch:=Sysfputc(TO_INT(@StringOut), Fp); (* Written character *)
            StringOut:=StringOut+1; (* String output pointer *)
        END_FOR;
    END_IF;
END_IF;
```

7.5.8 SysSpyData, system spy data

Type	Library
Function	XTarget_11_0

Questa funzione permette di inviare dati alla console di spionaggio (Accessibile da Telnet con il comando ***SpyData***). É possibile definire sia il modo di visualizzazione dei dati spiati ***Mode*** che abbinare una etichetta nella stringa spiata ***Label***.

Con il parametro ***TFlags*** è possibile definire un pattern di 32 bits che viene usato come trigger per la visualizzazione dei dati spiati nella console di spionaggio Telnet.

Eseguendo la funzione con tutti i parametri a "0" viene eseguito il controllo sullo spazio nella console di spionaggio Telnet. La funzione ritorna TRUE se vi è spazio per salvare il record spiato e FALSE se la console è occupata.

Parametri funzione:

Mode (USINT) Definisce il modo di visualizzazione dei dati spiati, ***Spy mode***.

TFlags (UDINT) Definisce i bits di trigger per la visualizzazione dei dati spiati.

Label (@USINT) Etichetta riportata nella stringa spiata.

DPtr (@USINT) Puntatore alla stringa dati da spiare.

La funzione ritorna:

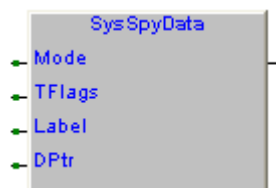
(BOOL) ***TRUE:*** Spionaggio possibile. ***FALSE:*** Console di spionaggio occupata.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9950100 Funzione eseguita in task fast o slow.

9950200 Errore allocazione memoria funzione.

9950990 Non implementata nel simulatore.

Esempi

Ecco un esempio di programma che invia ad ogni secondo tre records alla console di spionaggio.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	TString	STRING	Auto	[32]		..	Text string
2	SpyPulse	BOOL	Auto	No	FALSE	..	Spy data pulse
3	ABf	BOOL	Auto	No	FALSE	..	Auxiliary buffer

Esempio ST *(PTP116A400, ST_SysSpyData)*

```
(* Init the string to be spied and check if there is space to spy. *)

TString:='Hello!$r$n'; (* Text string *)
IF NOT(SysSpyData(0, 0, NULL, NULL))THEN RETURN; END_IF;

(* Check if is a second pulse. *)

IF (SysClock1000 = SpyPulse) THEN RETURN; END_IF;
SpyPulse:=SysClock1000; (* Spy data pulse *)

(* Send 3 spy data records. *)

ABf:=SysSpyData(SPY_ASCII, 16#00000001, ADR('Sp 1'), ADR(TString));
ABf:=SysSpyData(SPY_ASCHEX, 16#00000010, ADR('Sp 2'), ADR(TString));
ABf:=SysSpyData(SPY_BINARY+8, 16#00000100, ADR('Sp 3'), ADR(TString));
```

Console di spionaggio

Per attivare la console di spionaggio occorre accedere al sistema in Telnet, (Fare riferimento al Manuale riferimento comandi Telnet CPU SlimLine). Con il comando ***SpyData*** si attiva la console di spionaggio e sono visualizzati i vari records dati.

Toolly by ELSIST Srl - [Terminal]

File Utilities Devices Window Help

```
Elsist SlimLine (SFW184A80-), maintenance shell
Code: MPS046B200, Serial:1
Login: Admin
Password: *****

[Admin]> SpyData
Spy data active, type "Ctrl-C" to exit...
15:10:24(9999)|Sp 1|Hello!..
15:10:24(0000)|Sp 2|Hello![0D][0A]
15:10:24(0000)|Sp 3|48 65 6C 6C 6F 21 0D 0A
15:10:25(0999)|Sp 1|Hello!..
15:10:25(0000)|Sp 2|Hello![0D][0A]
15:10:25(0000)|Sp 3|48 65 6C 6C 6F 21 0D 0A
15:10:26(0999)|Sp 1|Hello!..
15:10:26(0000)|Sp 2|Hello![0D][0A]
15:10:26(0000)|Sp 3|48 65 6C 6C 6F 21 0D 0A
15:10:27(0999)|Sp 1|Hello!..
15:10:27(0000)|Sp 2|Hello![0D][0A]
15:10:27(0000)|Sp 3|48 65 6C 6C 6F 21 0D 0A
```

Udp Client Server | Serial | TcpIp Client | TcpIp Server

192.168.0.162 IP

23 Port

Stop | Config | Clear | Send file

Terminal: ASCII | Tx | Rx

[31/10/2013 15:11:46] Disconnection: 192.168.0.162 : 23
[31/10/2013 15:11:46] Stop
[31/10/2013 15:11:55] Connection: 192.168.0.162 : 23
[31/10/2013 15:11:55] Start

Commands | Timed Commands

Load Commands | Save Commands

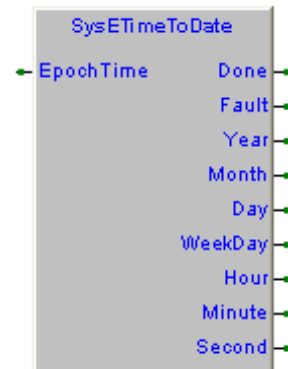
Admin$0D H Cmd 1

SpyData$0D H Cmd 2

7.6 Funzioni ed FB per gestione Data/Ora

7.6.1 SysETimeToDate, epoch time to date conversion

Type	Library
FB	XTarget_07_0

Questo blocco funzione esegue la conversione della data espressa in epoch time. Occorre fornire al blocco funzione la data espressa nel formato epoch ime come presente nella variabile di sistema ***SysDateTime***, in uscita dal blocco funzione avremo i valori di data espressi nel formato Giorno/Mese/Anno ed Ora:Minuti:Secondi.

EpochTime (UDINT) Occorre specificare la data espressa in epoch time.

Done (BOOL) Attivato al termine della conversione.

Fault (BOOL) Errore di conversione, viene attivato in caso di errore nella conversione.

Year (UINT) Ritorna il valore di anno (Range da 1970 a 2099)

Month (USINT) Ritorna il valore di mese dell'anno (Range da 1 a 12)

Day (USINT) Ritorna il valore di giorno del mese (Range da 1 a 31)

WeekDay (USINT) Ritorna il valore di giorno della settimana (Range da 0 a 6)
0: Domenica, 1:Lunedì, 2:Martedì, 3:Mercoledì, 4:Giovedì, 5:Venerdì, 6:Sabato

Hour (USINT) Ritorna il valore di ora (Range da 0 a 23)

Minute (USINT) Ritorna il valore di minuti (Range da 0 a 59)

Second (USINT) Ritorna il valore di secondi (Range da 0 a 59)

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

9986050 Errore allocazione blocco funzione.

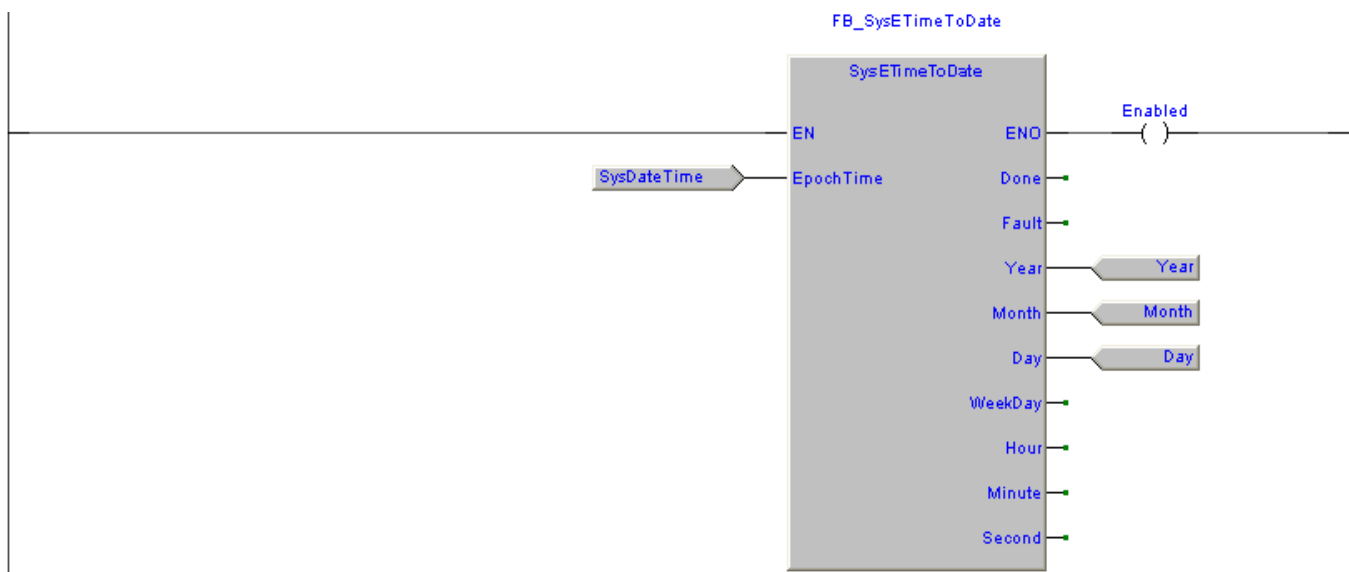
9986060 Errore versione blocco funzione.

Esempi

Viene convertito il valore di data ed ora espresso in epoch time dalla variabile ***SysDateTime*** e viene ritornato il valore di anno, mese e giorno nelle tre variabili definite.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FB_SysETimeToDate	SysETimeToDate	Auto	No	0	..	FB SysETimeToDate data
2	Year	UINT	Auto	No	0	..	Year
3	Month	USINT	Auto	No	0	..	Month
4	Day	USINT	Auto	No	0	..	Day

Esempio LD *(PTP116A100, LD_SysETimeToDate)*

Esempio IL *(PTP116A100, IL_SysETimeToDate)*

```
(* Transfer system date e time to FB input variable. *)

    LD SysDateTime
    ST FB_SysETimeToDate.EpochTime

    CAL FB_SysETimeToDate (* Call the SysETimeToDate function block *)

(* Transfer the FB output variables to program variables. *)

    LD FB_SysETimeToDate.Year
    ST Year

    LD FB_SysETimeToDate.Month
    ST Month

    LD FB_SysETimeToDate.Day
    ST Day
```

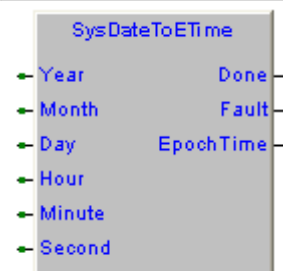
Esempio ST *(PTP116A100, ST_SysETimeToDate)*

```
(* Here FB SysETimeToDate is executed and variables copied. *)

    FB_SysETimeToDate.EpochTime:=SysDateTime;
    FB_SysETimeToDate();
    Year:=FB_SysETimeToDate.Year; (* Year *)
    Month:=FB_SysETimeToDate.Month; (* Month *)
    Day:=FB_SysETimeToDate.Day; (* Day *)
```

7.6.2 SysDateToETime, date to epoch time conversion

Type	Library
FB	XTarget_07_0

Questo blocco funzione esegue la conversione della data-ora in epoch time. Occorre fornire al blocco funzione la data e l'ora ed in uscita dal blocco funzione avremo un valore in epoch time.

SysDateToETime
Year Done
Month Fault
Day EpochTime
Hour
Minute
Second

Year (UINT) Definisce il valore di anno (Range da 1970 a 2099)

Month (USINT) Definisce il valore di mese dell'anno (Range da 1 a 12)

Day (USINT) Definisce il valore di giorno del mese (Range da 1 a 31)

Hour (USINT) Definisce il valore di ora (Range da 0 a 23)

Minute (USINT) Definisce il valore di minuti (Range da 0 a 59)

Second (USINT) Definisce il valore di secondi (Range da 0 a 59)

Done (BOOL) Attivato al termine della conversione.

Fault (BOOL) Errore di conversione, viene attivato in caso di errore nella conversione.

EpochTime (UDINT) Ritorna data espressa in epoch time.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

9987050 Errore allocazione blocco funzione.

9987060 Errore versione blocco funzione.

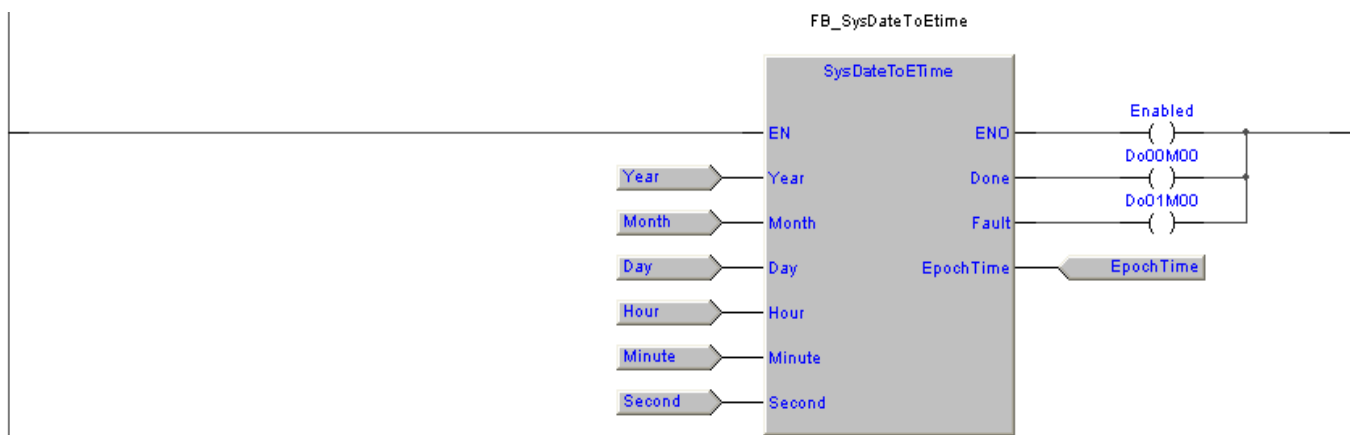
9987200 Errore durante l'esecuzione del blocco funzione.

Esempi

Viene convertito il valore di data ed ora in epoch time. Esempio definendo il valore di data 9/4/2010 e ora 14:20:15 avremo in uscita il valore 1270822815.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FB_SysDateToEtime	SysDateToETime	Auto	No	0	..	FB SysDateToETime data
2	Hour	USINT	Auto	No	0	..	Day
3	Minute	USINT	Auto	No	0	..	Month
4	Second	USINT	Auto	No	0	..	Year
5	Day	USINT	Auto	No	0	..	Day
6	Month	USINT	Auto	No	0	..	Month
7	Year	UINT	Auto	No	0	..	Year
8	EpochTime	UDINT	Auto	No	0	..	Epoch time

Esempio LD *(PTP116A100, LD_SysDateToETime)*

Esempio IL *(PTP116A100, IL_SysDateToETime)*

```
(* Transfer date e time to FB input variable. *)

    LD Year
    ST FB_SysDateToEtime.Year

    LD Month
    ST FB_SysDateToEtime.Month

    LD Day
    ST FB_SysDateToEtime.Day

    CAL FB_SysDateToEtime (* Call the SysDateToEtime function block *)

(* Transfer the FB output variables to program variables. *)

    LD FB_SysDateToEtime.EpochTime
    ST EpochTime
```

Esempio ST *(PTP116A100, ST_SysDateToETime)*

```
(* Here FB SysDateToETime is executed and variables copied. *)

    FB_SysDateToEtime.Year:=Year;
    FB_SysDateToEtime.Month:=Month;
    FB_SysDateToEtime.Day:=Day;
     FB_SysDateToEtime(); (* Call the SysDateToEtime function block *)
    EpochTime:=FB_SysDateToEtime.EpochTime; (* Epoch time *)
```

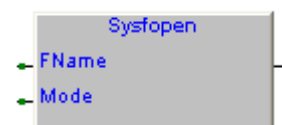
7.7 Funzioni ed FB per gestione terminale di I/O

7.7.1 Sysfopen, file open

Type	Library
Function	XTarget_07_0

Questa funzione permette l'apertura del collegamento tra la risorsa indicata dal parametro ***FName***, ed un flusso di dati ***stream*** da impiegare nelle successive chiamate alle funzioni di I/O. La funzione ritorna il pointer alla risorsa.

Se la risorsa indicata è già aperta oppure il nome della risorsa è errato, la funzione ritorna ***NULL***. Se si sta aprendo un file su disco per crearlo, accertarsi che il disco sia formattato.

Parametri funzione:

FName (STRING[20]) E' il nome della risorsa da utilizzare.

Name	Resource
COM0	Serial port COM0
COM1	Serial port COM1
COM2	Serial port COM2
PCOMx.y	Porta seriale y su modulo periferico con indirizzo x
UDPSKT	UDP socket
TCPSKT	TCP socket
pathname	Percorso completo comprensivo del nome file (es.: 'Storage/myFile.txt')

Mode (STRING[4]) Indica il modo in cui la risorsa è aperta: r=read; w=write ; a=append. Per le porte seriali definire 'rw'.
Per creare un file su disco, occorre eseguire l'apertura in 'w' o 'a'.
L'apertura in 'w' su un file esistente, provoca la cancellazione del contenuto.
L'apertura in 'r' o 'w' posizionano l'indicatore di posizione dello stream all'inizio del file, l'apertura in 'a' lo posiziona alla fine.

La funzione ritorna:

(FILEP) Pointer alla risorsa.
NULL: In caso di errore.

Codici di errore

In caso di errore la funzione torna con ***NULL*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9996100 Nome risorsa ***FName*** ha lunghezza errata.

9996110 Nome risorsa ***FName*** ha lunghezza errata.

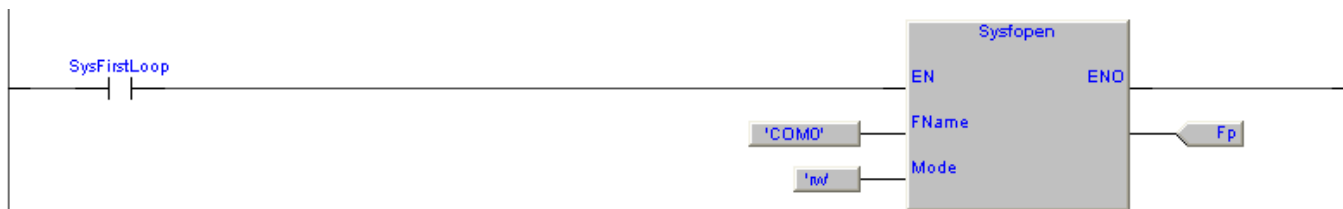
9996200~2 Impossibile utilizzare porta da programma utente.

9996990 Non implementata nel simulatore.

Esempi

Viene aperta la porta seriale in read/write.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer

Esempio LD

Esempio ST *(PTP116A100, ST_Sysfopen)*

```
IF (Fp = NULL) THEN
    Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
END_IF;
```

7.7.2 Sysfclose, file close

Type	Library
Function	XTarget_07_0

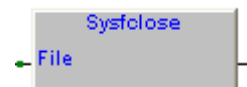
Questa funzione permette la chiusura del collegamento alla risorsa indicata dal parametro ***File***, precedentemente aperto dalla funzione Sysfopen.

In caso di errore chiusura, la funzione ritorna ***EOF***.

Parametri funzione:

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(INT) ***0:*** Se esecuzione corretta.
EOF: In caso di errore.

Codici di errore

In caso di errore la funzione torna con ***EOF*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9973100 Terminale di I/O usato in task fast o slow.

9973200 Errore nella chiusura della risorsa.

9973990 Non implementata nel simulatore.

Esempi

Viene aperta e successivamente chiusa la porta seriale ***COM0***. Se la porta è correttamente aperta viene attivata l'uscita ***Do00M00***. Se la porta è correttamente chiusa viene attivata l'uscita ***Do01M00***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer

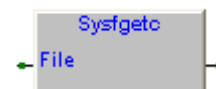
Esempio ST *(PTP116A100, ST_Sysfclose)*

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
        Do00M00:=(Fp <> NULL); (* Output is set if port is opened *)
    END_IF;

(* Here the COM0 port is closed. *)

    IF (Fp <> NULL) THEN
        Do01M00:=(Sysfclose(Fp) <> EOF); (* Output is set if port is closed *)
    END_IF;
```

7.7.3 Sysfgetc, get character from file

Type	Library
Function	XTarget_07_0

Questa funzione ritorna un carattere dal flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione Sysfopen.

La funzione ritorna il carattere letto dallo stream. In caso di errore o se nessun dato dallo stream, la funzione ritorna ***EOF***. Per essere certi che vi siano caratteri dallo stream è possibile utilizzare la funzione SysGetIChars che ne ritorna il numero.

Parametri funzione:

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(INT) Carattere letto dal flusso di dati.
EOF: In caso di errore o se nessun dato dallo stream.

Codici di errore

In caso di errore la funzione torna con ***EOF*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9972100 Terminale di I/O usato in task fast o slow.

9972990 Non implementata nel simulatore.

Esempi

Viene eseguita apertura porta seriale ***COM0*** e controllato se caratteri disponibili dalla porta. Se almeno un carattere è disponibile ne viene eseguita lettura ed il carattere letto è trasferito nella variabile ***Ch***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Ch	INT	Auto	No	0	..	Character read

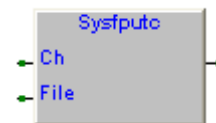
Esempio ST *(PTP116A100, ST_Sysfgetc)*

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

(* Here check if a character is available from port and read it. *)

    IF (Fp <> NULL) THEN
        IF (TO_BOOL(SysGetIChars(Fp))) THEN
            Ch:=Sysfgetc(Fp); (* Get input character *)
        END_IF;
     END_IF;
```

7.7.4 Sysfputc, put character to file

Type	Library
Function	XTarget_07_0

Questa funzione invia un carattere sul flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione Sysfopen.

La funzione ritorna il carattere scritto sullo stream. In caso di errore o se lo stream non accetta il dato, la funzione ritorna EOF. Per essere certi che vi sia spazio sullo stream per accettare il carattere, è possibile utilizzare la funzione SysGetOSpace che ritorna lo spazio disponibile.

Parametri funzione:

Ch (INT) Carattere da inviare sul flusso dati.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(INT) Carattere scritto sul flusso di dati. ***EOF:*** In caso di errore o se lo stream non accetta il dato.

Codici di errore

In caso di errore la funzione torna con ***EOF*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9971100 Terminale di I/O usato in task fast o slow.

9971990 Non implementata nel simulatore.

Esempi

E' riportato un semplice programma che esegue l'eco dei caratteri ricevuti dalla porta seriale ***COM0***. Viene eseguita apertura porta seriale ***COM0*** e controllato se caratteri disponibili dalla porta. Se almeno un carattere è disponibile ne viene eseguita lettura e successiva ritrasmissione.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Ch	INT	Auto	No	0	..	Character read

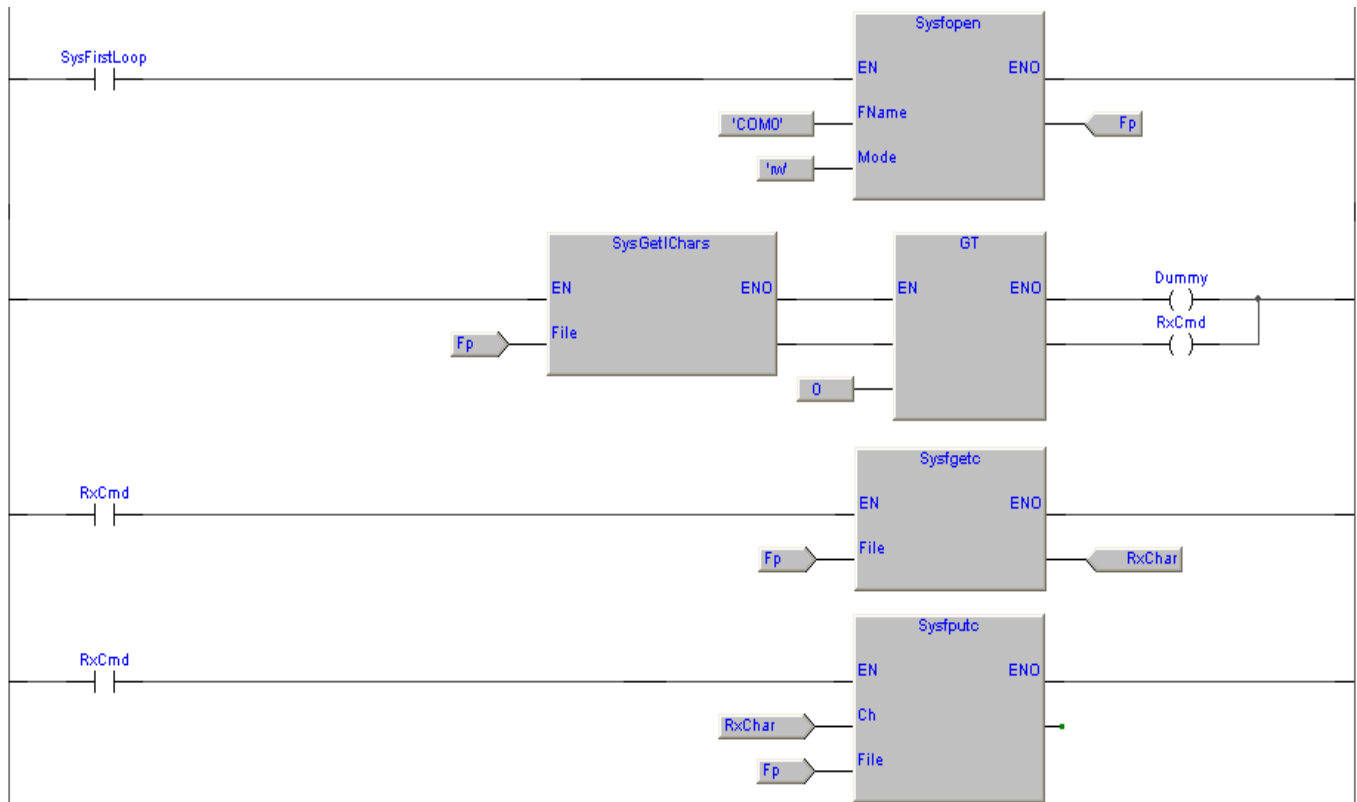
Esempio ST *(PTP116A100, ST_Sysfputc)*

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

(* Here execute the received characters echo. *)

    IF (Fp <> NULL) THEN
        IF (TO_BOOL(SysGetIChars(Fp))) AND (TO_BOOL(SysGetOSpace(Fp))) THEN
            Ch:=Sysfgetc(Fp); (* Get input character *)
            Ch:=Sysfputc(Ch, Fp); (* Put input character *)
        END_IF;
    END_IF;
```

Utilizzando le funzioni di gestione terminale di I/O è possibile realizzare un semplice programma che esegue l'echo del carattere ricevuto dalla porta seriale ***COM0***. La porta viene aperta con il modo impostato di default (115200, e, 8, 1), se è ricevuto un carattere dalla porta si attiva ***RxCmd*** ed il carattere ricevuto è copiato in ***RxChar***. La ricezione di un carattere provoca la trasmissione del carattere ricevuto.

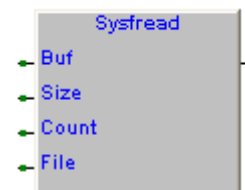
Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
3	RxChar	INT	Auto	No	0	..	Received char
4	RxCmd	BOOL	Auto	No	FALSE	..	Rx command

Esempio LD *(PTP19A000, SerialEcho)*

Type	Library
Function	XTarget_07_0

7.7.5 Sysfread, read data from file

Questa funzione esegue la lettura di un numero definito di stringhe di lunghezza definita dal flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione Sysfopen.

La funzione ritorna il numero di stringhe dati lette. Se nello ***stream*** non ci sono abbastanza stringhe da soddisfare i parametri, viene ritornato un numero minore di stringhe lette rispetto al valore definito.

Parametri funzione:

Buf (@STRING) Indirizzo della stringa dove trasferire le stringhe lette.

Size (INT) Lunghezza in caratteri delle stringhe da leggere.

Count (INT) Numero di stringhe da leggere.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(INT) Numero di stringhe lette, se il valore ritornato è minore di ***Count***, significa che non vi erano abbastanza dati nello stream.

Codici di errore

In caso di errore la funzione torna con ***0*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9970100 Terminale di I/O usato in task fast o slow.

9970990 Non implementata nel simulatore.

Esempi

Vengono attesi almeno 5 caratteri ricevuti dalla porta seriale e quando ricevuti viene letta una stringa di 5 caratteri (5 stringhe di 1 carattere), la stringa letta è trasferita nella variabile ***RxString***. La stringa letta viene poi ritrasmessa sulla porta seriale, notare come anche nella trasmissione è trasmessa una stringa di 5 caratteri (1 stringa di 5 caratteri) .

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	RxString	STRING	Auto	[10]	0	..	Received string
3	RxChars	INT	Auto	No	0	..	Received characters
4	TxChars	INT	Auto	No	0	..	Transmitted characters

Esempio ST *(PTP116A100, ST_Sysfread)*

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

(* Here wait until at least 5 chars are received and echoes them. *)

    IF (Fp <> NULL) THEN
        IF (SysGetIChars(Fp) >= 5) THEN
            RxChars:=Sysfread(ADR(RxString), 1, 5, Fp); (* Received characters *)
            TxChars:=Sysfread(ADR(RxString), 5, 1, Fp); (* Received characters *)
        END_IF;
    END_IF;
```

7.7.6 Sysfwrite, write data to file

Type	Library
Function	XTarget_07_0

Questa funzione esegue la scrittura di un numero definito di stringhe di lunghezza definita nel flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione Sysfopen.

La funzione ritorna il numero di stringhe dati scritte. Se nello stream non c'è abbastanza spazio per contenere il numero di stringhe definito, viene ritornato un numero minore di stringhe scritte rispetto al valore definito.

Parametri funzione:

Buf (@STRING) Indirizzo della stringa da scrivere.

Size (INT) Lunghezza in caratteri delle stringhe da scrivere.

Count (INT) Numero di stringhe da scrivere.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(INT) Numero di stringhe scritte, se valore ritornato minore di ***Count***, non vi era abbastanza spazio nello stream.

Codici di errore

In caso di errore la funzione torna con ***0*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9969100 Terminale di I/O usato in task fast o slow.

9969990 Non implementata nel simulatore.

Esempi

Vengono attesi almeno 5 caratteri ricevuti dalla porta seriale e quando ricevuti viene letta una stringa di 5 caratteri (5 stringhe di 1 carattere), la stringa letta è trasferita nella variabile ***RxString***. La stringa letta viene poi ritrasmessa sulla porta seriale, notare come anche nella trasmissione è trasmessa una stringa di 5 caratteri (1 stringa di 5 caratteri) .

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	RxString	STRING	Auto	[10]	0	..	Received string
3	RxChars	INT	Auto	No	0	..	Received characters
4	TxChars	INT	Auto	No	0	..	Transmitted characters

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

(* Here wait until at least 5 chars are received and echoes them. *)

    IF (Fp <> NULL) THEN
        IF (SysGetIChars(Fp) >= 5) THEN
            RxChars:=Sysfread(ADR(RxString), 1, 5, Fp); (* Received characters *)
            TxChars:=Sysfwrite(ADR(RxString), 5, 1, Fp); (* Transmitted characters *)
        END_IF;
    END_IF;
```

7.7.7 SysGetIChars, get input available characters from file

Type	Library
Function	XTarget_07_0

SysGetIChars
File

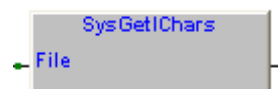
Questa funzione ritorna il numero di caratteri disponibili per la lettura dal flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione Sysfopen.

Se il valore ritornato è diverso da ***0*** i caratteri potranno essere letti con la funzione Sysfgetc.

Parametri funzione:

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(INT) Numero di caratteri disponibili dal flusso dati.

Codici di errore

In caso di errore la funzione torna con ***0*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9968100 Terminale di I/O usato in task fast o slow.

9968990 Non implementata nel simulatore.

Esempi

E' riportato un semplice programma che esegue l'eco dei caratteri ricevuti dalla porta seriale ***COM0***. Viene eseguita apertura porta seriale ***COM0*** e controllato se caratteri disponibili dalla porta. Se almeno un carattere è disponibile ne viene eseguita lettura e successiva ritrasmissione.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Ch	INT	Auto	No	0	..	Character read

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

(* Here execute the received characters echo. *)

    IF (Fp <> NULL) THEN
        IF (TO_BOOL(SysGetIChars(Fp))) AND (TO_BOOL(SysGetOSpace(Fp))) THEN
            Ch:=Sysfgetc(Fp); (* Get input character *)
            Ch:=Sysfputc(Ch, Fp); (* Put input character *)
        END_IF;
    END_IF;
```

Type	Library
Function	XTarget_07_0

7.7.8 SysGetOSpace, get output available space on file

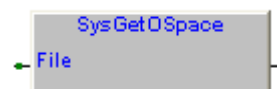
Questa funzione ritorna lo spazio disponibile per la scrittura dati sul flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

Se il valore ritornato è diverso da ***0*** i caratteri potranno essere scritti con la funzione ***Sysfputc***.

Parametri funzione:

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(INT) Spazio disponibile sul flusso dati per trasferire caratteri.
Se buffer vuoto viene ritornata la dimensione del buffer di trasmissione.

Codici di errore

In caso di errore la funzione torna con ***0*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9967100 Terminale di I/O usato in task fast o slow.

9967990 Non implementata nel simulatore.

Esempi

E' riportato un semplice programma che esegue l'eco dei caratteri ricevuti dalla porta seriale ***COM0***. Viene eseguita apertura porta seriale ***COM0*** e controllato se caratteri disponibili dalla porta. Se almeno un carattere è disponibile ne viene eseguita lettura e successiva ritrasmissione.

Definizione variabili

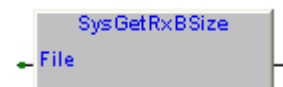
	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Ch	INT	Auto	No	0	..	Character read

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

(* Here execute the received characters echo. *)

    IF (Fp <> NULL) THEN
        IF (TO_BOOL(SysGetIChars(Fp))) AND (TO_BOOL(SysGetOSpace(Fp))) THEN
            Ch:=Sysfgetc(Fp); (* Get input character *)
            Ch:=Sysfputc(Ch, Fp); (* Put input character *)
        END_IF;
    END_IF;
```

7.7.9 SysGetRxBSize, get file Rx input buffer size

Type	Library
Function	XTarget_07_0

Questa funzione ritorna la dimensione del buffer di input (Ricezione) sul flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

Parametri funzione:

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(UDINT) Dimensione buffer di input espressa in numero di caratteri (Bytes).

Codici di errore

In caso di errore la funzione torna con ***0*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9966100 Terminale di I/O usato in task fast o slow.

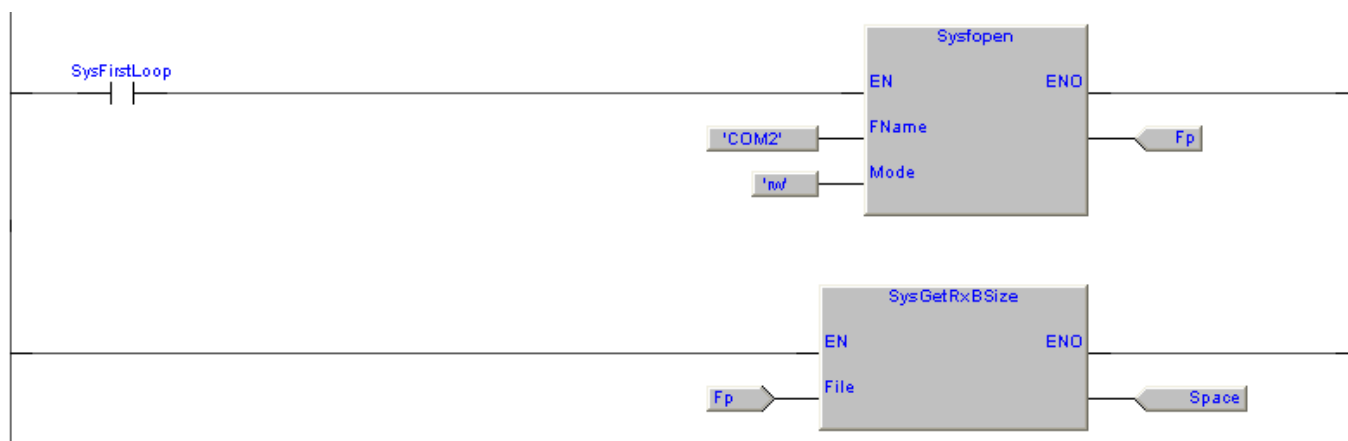
9966990 Non implementata nel simulatore.

Esempi

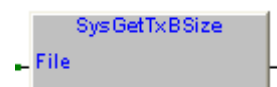
E' riportato un semplice programma che ritorna la dimensione del buffer di input (Ricezione) della porta seriale ***COM2***. Il valore ritornato espresso in numero di caratteri (Bytes), è trasferito nella variabile ***Space***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	Space	UDINT	Auto	No	0	..	Rx buffer space (Nr of Chars)

Esempio LD

7.7.10 SysGetTxBSize, get file Tx output buffer size

Type	Library
Function	XTarget_07_0

Questa funzione ritorna la dimensione del buffer di output (Trasmissione) sul flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

Parametri funzione:

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(UDINT) Dimensione buffer di output espressa in numero di caratteri (Bytes).

Codici di errore

In caso di errore la funzione torna con ***0*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9965100 Terminale di I/O usato in task fast o slow.

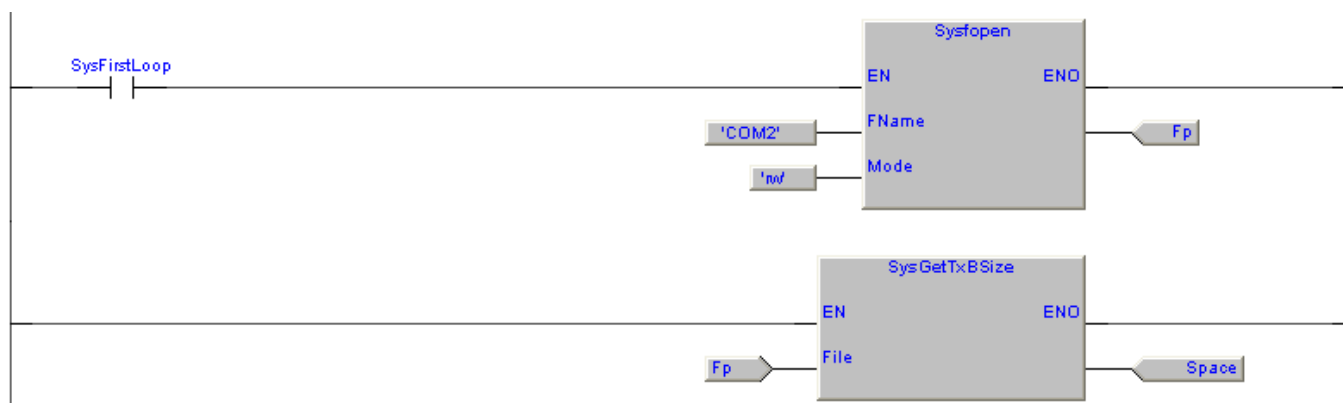
9965990 Non implementata nel simulatore.

Esempi

E' riportato un semplice programma che ritorna la dimensione del buffer di output (Trasmissione) della porta seriale ***COM2***. Il valore ritornato espresso in numero di caratteri (Bytes), è trasferito nella variabile ***Space***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	Space	UDINT	Auto	No	0	..	Rx buffer space (Nr of Chars)

Esempio LD

7.7.11 SysFIBfClear, file input buffer clear

Type	Library
Function	XTarget_07_0

Questa funzione elimina tutti i caratteri in lettura presenti sul flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

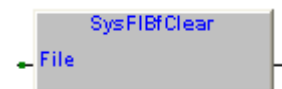
SysFIBfClear
File

La funzione ritorna ***FALSE*** in caso di errore.

Parametri funzione:

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(BOOL) ***FALSE:*** Errore esecuzione.
TRUE: Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9964100 Terminale di I/O usato in task fast o slow.

9964990 Non implementata nel simulatore.

Esempi

Se è attivo l'ingresso ***Di00M00*** tutti i caratteri in ingresso dalla porta seriale saranno cancellati e l'uscita ***Do00M00*** attivata.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

(* If the input is active the input buffer is cleared. *)

    IF (Fp <> NULL) THEN
        IF (Di00M00) THEN Do00M00:=SysFIBfClear(Fp); END_IF;
    END_IF;
```

7.7.12 SysFOBfClear, file output buffer clear

Type	Library
Function	XTarget_07_0

SysFOBfClear
File

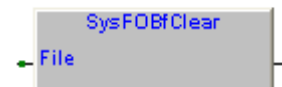
Questa funzione elimina tutti i caratteri in uscita presenti sul flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

La funzione ritorna ***FALSE*** in caso di errore.

Parametri funzione:

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(BOOL) ***FALSE:*** Errore esecuzione.
TRUE: Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9963100 Terminale di I/O usato in task fast o slow.

9963990 Non implementata nel simulatore.

Esempi

Se è attivo l'ingresso ***Di00M00*** tutti i caratteri in uscita dalla porta seriale saranno cancellati e l'uscita ***Do00M00*** attivata.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

(* If the input is active the ouput buffer is cleared. *)

    IF (Fp <> NULL) THEN
        IF (Di00M00) THEN Do00M00:=SysFOBfClear(Fp); END_IF;
    END_IF;
```

7.7.13 SysFOBfFlush, file output buffer flush

Type	Library
Function	XTarget_07_0

Questa funzione forza l'uscita immediata dei caratteri presenti sul flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***, sulla risorsa connessa.

SysFOBfFlush
File

La funzione ritorna ***FALSE*** in caso di errore.

Parametri funzione:

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(BOOL) ***FALSE:*** Errore esecuzione.
TRUE: Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9962100 Terminale di I/O usato in task fast o slow.

9962990 Non implementata nel simulatore.

Esempi

Se è attivo l'ingresso ***Di00M00*** tutti i caratteri presenti nel buffer di uscita della porta seriale saranno trasmessi e l'uscita ***Do00M00*** attivata.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

(* If the input is active the ouput buffer is cleared. *)

    IF (Fp <> NULL) THEN
        IF (Di00M00) THEN Do00M00:=SysFOBfFlush(Fp); END_IF;
    END_IF;
```

7.7.14 SysVarfprintf, variable print to file

Type	Library
Function	XTarget_07_0

SysVarfprintf
File
Format
VarType
VarAdd

Questa funzione esegue la stampa formattata di una variabile sullo stream collegato al parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

La stringa ***Format*** specifica il formato con il quale stampare la variabile. Mentre in ***VarType*** è indicato il tipo di variabile ed in ***VarAdd*** il suo indirizzo.

La funzione ritorna il numero di caratteri trasferiti nello stream, ***EOF*** in caso di errore.

Parametri funzione:

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

Format (STRING[80]) Ha due tipi di argomenti, i caratteri ordinari che vengono copiati nello stream di uscita, e le specifiche di conversione, contraddistinte dal simbolo percentuale (%) e da un carattere che specifica il formato con il quale stampare la variabile definita.

VarType (USINT) Tipo variabile, come indicato nella tabella ***Variable types definition***.

VarAdd (UDINT) Indirizzo variabile.

La funzione ritorna:

(INT) Numero di caratteri trasferiti nello stream. ***EOF:*** Errore esecuzione.

Codici di errore

In caso di errore la funzione torna con ***EOF*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9998010 Valore di ***File*** non definito.

9968100 Terminale di I/O usato in task fast o slow.

9998200 Tipo variabile non gestito, controllare ***VarType***.

9998990 Non implementata nel simulatore.

Esempi

Su fronte attivazione ingresso ***Di00M00*** viene incrementata la variabile ***Counter*** ed il suo valore inviato sulla porta seriale ***COM0***. Nella variabile ***NrOfChars*** viene caricato il numero di caratteri inviati in uscita sulla porta seriale.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag
3	Counter	UDINT	Auto	No	0	..	Counter
4	NrOfChars	INT	Auto	No	0	..	

Esempio ST *(PTP116A300, ST_SysVarfprintf)*

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN Fp:=Sysfopen('COM0', 'rw'); END_IF;

    IF (Fp <> NULL) THEN

        IF (Di00M00 <> Pulse) THEN
            Pulse:=Di00M00; (* Pulse flag *)

            IF (Di00M00) THEN
                Counter:=Counter+1; (* Counter *)
                NrOfChars:=SysVarfprintf(Fp, 'Counter:%04d$r$n', UDINT_TYPE, ADR(Counter));
            END_IF;
        END_IF;
    END_IF;
```

7.8 File system

Le “CPU SlimLine ARM7” a partire dal firmware versione ***SFW167C100***, possono gestire il file system. In tali CPU esistono due directories predefinite:

Storage: Directory allocata sulla memoria EEPROM presente su SlimLine (Tutte le versioni).

SDCard: Directory allocata sul chip SD Card che deve essere inserito nell'apposito connettore.

Per le operazioni di formattazione del file system si rimanda al manuale utente, il file system è raggiungibile da FTP, quindi utilizzando un client FTP è possibile creare nuovi files, cancellare files esistenti, rinominare files esistenti, leggere e scrivere dati nei files.

7.8.1 Sysremove, file remove

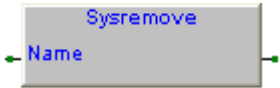
Type	Library
Function	XTarget_07_0

Questa funzione esegue la rimozione (cancellazione) di un file. In ***Name*** occorre definire il nome del file da eliminare specificando l'intero percorso (Esempio Storage/File.txt).

Se l'operazione di rimozione va a buon fine la funzione ritorna ***TRUE***, in caso di errore viene ritornato ***FALSE***.

Parametri funzione:

Name (STRING[32]) Nome del file da cancellare.

La funzione ritorna:

(BOOL) ***FALSE:*** Errore esecuzione.
TRUE: Ok esecuzione.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9961100 Funzione richiamata in task fast o slow.

9961150 Errore nella dichiarazione del nome file.

9961160 Directory non accessibile da utente “Admin”.

9961200 Errore nella cancellazione del file.

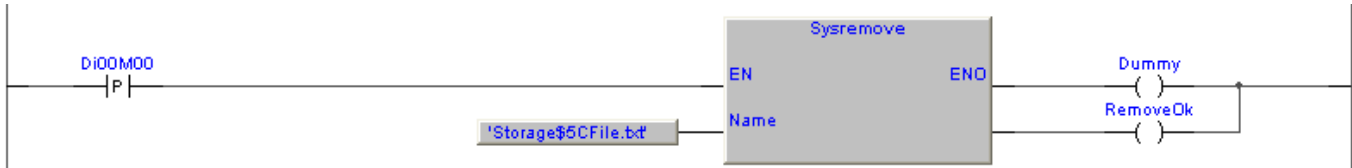
9961990 Non implementata nel simulatore.

Esempi

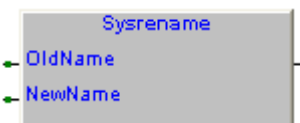
Sul fronte di attivazione dell'ingresso digitale ***Di00M00*** viene eliminato il file ***File.txt*** presente nella directory ***Storage***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RemoveOk	BOOL	Auto	No	FALSE	..	File remove Ok
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy

Esempio LD

7.8.2 Sysrename, file rename

Type	Library
Function	XTarget_07_0

Questa funzione esegue il cambiamento del nome di un file. In ***OldName*** occorre definire il nome del file da rinominare specificando l'intero percorso (Esempio Storage/OldFile.txt), in ***NewName*** occorre definire il nuovo nome del file specificando l'intero percorso (Esempio Storage/NewFile.txt).

Se l'operazione di rinomina va a buon fine la funzione ritorna ***TRUE***, in caso di errore viene ritornato ***FALSE***.

Parametri funzione:

OldName (STRING[32]) Nome del file da rinominare.

NewName (STRING[32]) Nuovo nome da dare al file.

La funzione ritorna:

(BOOL) ***FALSE:*** Errore esecuzione.
TRUE: Ok esecuzione.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9960100 Funzione richiamata in task fast o slow.

9960150 Errore nella dichiarazione ***OldName***.

9960160 Directory file ***OldName*** non accessibile da utente “Admin”.

9960170 Errore nella dichiarazione ***NewName***.

9960180 Directory file ***NewName*** non accessibile da utente “Admin”.

9960200 Errore nella rinominazione del file.

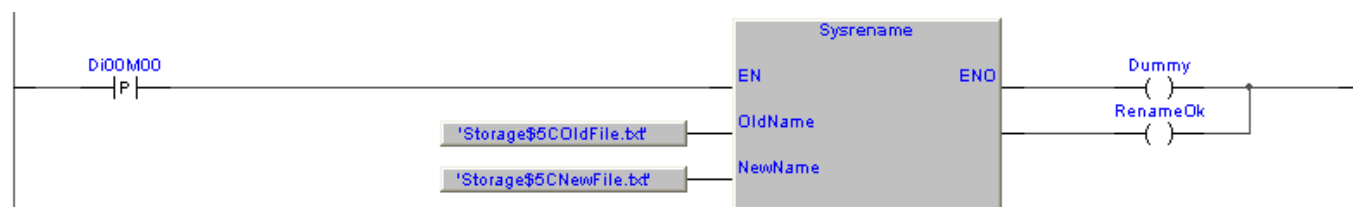
9960990 Non implementata nel simulatore.

Esempi

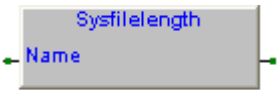
Sul fronte di attivazione dell'ingresso digitale ***Di00M00*** viene rinominato il file ***OldFile.txt*** presente nella directory ***Storage***. Il file assumerà il nuovo nome di ***NewFile.txt***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Dummy	BOOL	Auto	No	FALSE	..	Dummy
2	RenameOk	BOOL	Auto	No	FALSE	..	File rename Ok

Esempio LD

7.8.3 Sysfilelength, file lengh

Type	Library
Function	XTarget_07_0

Questa funzione ritorna la lunghezza in bytes di un file. In ***Name*** occorre definire il nome del file di cui si vuole conoscere la lunghezza specificando l'intero percorso (Esempio Storage/File.txt).

Se il file indicato non è presente, la funzione ritorna ***-1***.

Parametri funzione:

Name (STRING[32]) Nome del file di cui si vuole conoscere la lunghezza.

La funzione ritorna:

(DINT) Lunghezza file (Bytes). ***EOF*** se file non presente.

Codici di errore

In caso di errore la funzione torna ***EOF*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9959100 Funzione richiamata in task fast o slow.

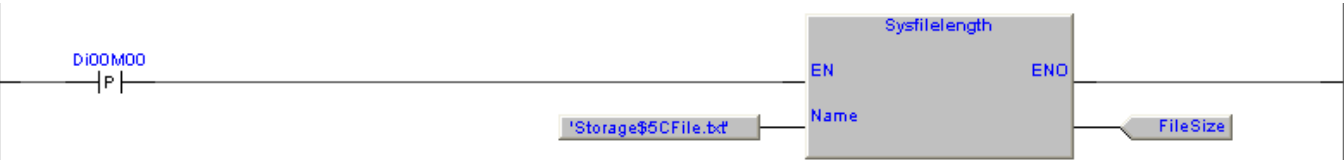
9959990 Non implementata nel simulatore.

Esempi

Sul fronte di attivazione dell'ingresso digitale ***Di00M00*** viene ritornata la lunghezza del file ***File.txt*** presente nella directory ***Storage***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FileSize	DINT	Auto	No	0	..	File size

Esempio LD

7.8.4 Sysfseek, file seek

Type	Library
Function	XTarget_07_0

Questa funzione permette di cambiare l'indicatore di posizione dello stream collegato al parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

Offset specifica il numero di bytes dall'origine dove andrebbe posizionato l'indicatore di posizione. ***Origin*** specifica la posizione di origine rispetto alla quale spostare l'indicatore di posizione.

La funzione ritorna il valore attuale dell'indicatore di posizione. In caso di errore di posizionamento, l'indicatore di posizione rimane inalterato e la funzione ritorna ***EOF***.

Parametri funzione:

File (FILEP) — Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

Offset (DINT) — Numero di bytes dall'origine dove posizionare l'indicatore di posizione

Origin (INT) — Occorre specificare la posizione di origine, ***FSeek origin definition***.

La funzione ritorna:

(INT) — Valore attuale dell'indicatore di posizione. ***EOF*** se errore.

Codici di errore

In caso di errore la funzione torna ***EOF*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9958100 Funzione richiamata in task fast o slow.

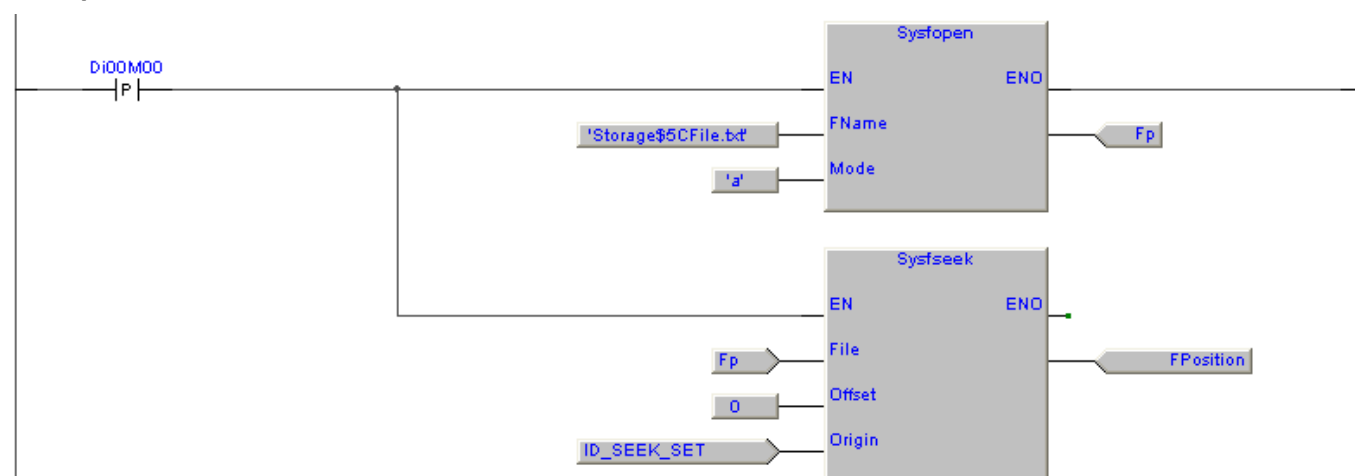
9958990 Non implementata nel simulatore.

Esempi

Sul fronte di attivazione dell'ingresso ***Di00M00*** viene posizionato l'indicatore di posizione all'inizio del file ***File.txt*** presente nella directory ***Storage***.

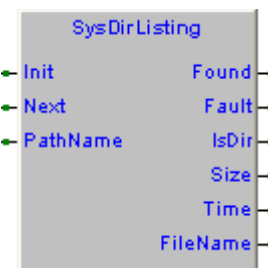
Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	FPosition	DINT	Auto	No	0	..	File position

Esempio LD

7.8.5 SysDirListing, directory listing

Type	Library
FB	XTarget_10_0

Questo blocco funzione esegue il listing di tutti i files contenuti in una directory, occorre passare il nome della directory di cui eseguire il listing in ***PathName***. Attivando l'ingresso ***Init*** si inizializza la lista dei files e viene ritornato il primo file trovato. Ad ogni comando ***Next*** si esegue la ricerca di un nuovo file nella directory selezionata.

Se file trovato l'uscita ***Found*** si attiva per un loop e ne viene rtornato il nome in ***FileName***. Terminato l'elenco di tutti i files presenti nella directory su comando ***Next*** non viene più attivata l'uscita ***Found*** e ***FileName*** è abblencato. L'uscita ***IsDir*** si attiva se il nome del file ritornato è quello di una sottodirectory.

Init (BOOL) Alla attivazione viene inizializzato l'indice dei files nella directory indicata e viene ritornato il nome del primo file trovato.

Next (BOOL) Alla attivazione viene ritornato il nome del file puntato dall'indice nella directory indicata. L'indice si incrementa puntando il successivo file.

PathName (STING[32]) Definizione directory di cui eseguire il listing. E' possibile definire anche eventuale filtro di ricerca (Esempio Storage*.log).

Found (BOOL) Si attiva per un loop se su comando ***Init*** o ***Next*** è stato trovato un nuovo file da listare.

Fault (BOOL) Si attiva per un loop se errore esecuzione.

IsDir (BOOL) Attiva se il nome di file ritornato appartiene ad una sottodirectory.

Size (UDINT) Dimensione in bytes del file.

Time (UDINT) Data ultima modifica del file in Epoch time (UTC).

FileName (STING[16]) Nome del file comprensivo di eventuale estensione.

Codici di errore

In caso di errore si attiva l'uscita Fault, con ***SysGetLastError*** è possibile rilevare il codice di errore.

9952050 Errore allocazione blocco funzione.

9952060 Terminato spazio memoria rilocabile, non è possibile eseguire l''FB.

9952070 Errore versione blocco funzione.

9952100 FB richiamata in task fast o slow.

9952200 Errore esecuzione directory listing.

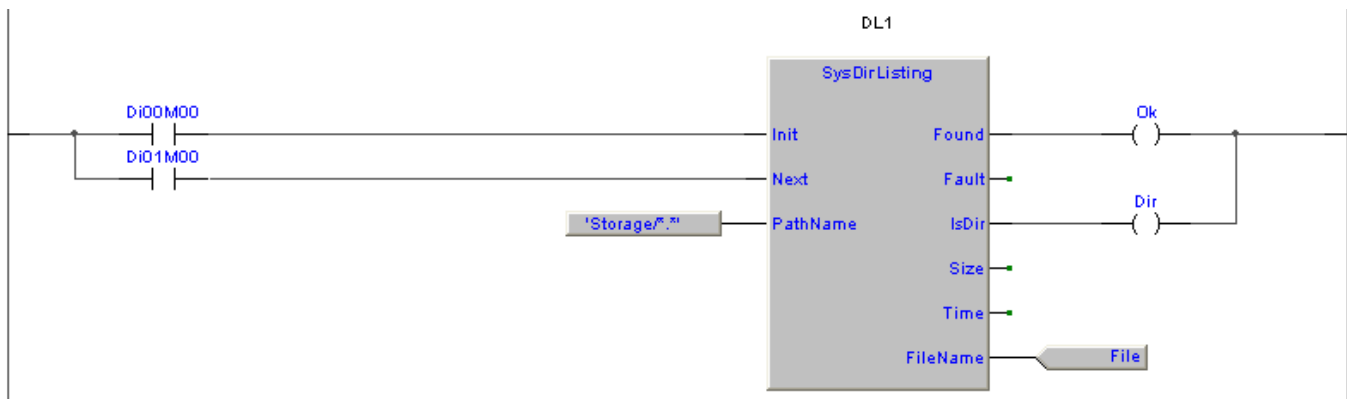
9952990 Non implementata nel simulatore.

Esempi

Sul fronte di attivazione dell'ingresso digitale ***Di00M00*** viene inizializzato il puntatore ai files presenti nella directory ***Storage***. Ad ogni attivazione dell'ingresso digitale ***Di01M00*** viene ritornato il nome del file presente nella directory. Se file presente si attiva per un loop ***Ok***, se si tratta di una sottodirectory si attiva per un loop ***Dir***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Dir	BOOL	Auto	No	FALSE	..	Is a directory
2	Ok	BOOL	Auto	No	FALSE	..	File found on Next command
3	File	STRING	Auto	[16]		..	File name
4	DL1	SysDirListing	Auto	No	0	..	FB directory listing

Esempio LD

7.9 Funzioni ed FB per gestione porta seriale

I sistemi SlimLine dispongono di più porte seriali in base alla versione del prodotto. Le porte seriali vengono identificate con un nome del tipo ***COMx*** dove al posto della ***x*** vi è il numero di porta seriale.

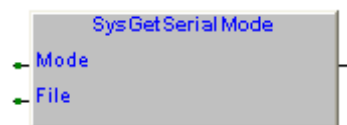
Per la corrispondenza tra il numero di porta ed il connettore fisico ad essa connesso, si rimanda al manuale hardware del prodotto.

7.9.1 SysGetSerialMode, get serial mode

Type	Library
Function	XTarget_07_0

SysGetSerialMode
Mode
File

Questa funzione ritorna il modo di comunicazione impostato sulla porta seriale connessa al parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

Nel parametro ***Mode*** occorre definire l'indirizzo della struttura ***SYSSERIALMODE*** in cui dovrà essere trasferito il modo seriale attualmente impostato sulla porta. La funzione ritorna ***FALSE*** in caso di errore.

Parametri funzione:

Mode (@SYSSERIALMODE) Indirizzo struttura ***SYSSERIALMODE*** in cui trasferire il modo letto.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(BOOL) ***FALSE:*** Errore esecuzione. ***TRUE:*** Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9995010 Valore di ***File*** non definito.

9995020 Indirizzo struttura ***SYSSERIALMODE*** non corretto, verificare ***Mode***.

9995100 ÷ 1 Errore esecuzione funzione.

9995990 Non implementata nel simulatore.

Esempi

Su fronte attivazione ingresso ***Di00M00*** viene salvato il modo impostato sulla porta seriale ***COM0*** nella variabile ***Sm*** e viene attivata l'uscita ***Do00M00***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Sm	SYSSERIALMODE	Auto	No	0	..	Serial mode data struct
3	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag

Esempio ST *(PTP116A300, ST_SysGetSerialMode)*

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

(* Check if the COM0 port is open. *)

    IF (Fp <> NULL) THEN

        (* Check if input is activated. *)

        IF (Di00M00 <> Pulse) THEN
            Pulse:=Di00M00; (* Pulse flag *)

            (* On input raising edge the serial mode is read. *)

            IF (Di00M00) THEN
                Do00M00:=SysGetSerialMode(ADR(Sm), Fp);
            END_IF;
        END_IF;
    END_IF;
```

Mettendo in watch la variabile ***Sm*** di tipo ***SYSSERIALMODE*** possiamo vedere i valori di tutti i suoi membri come riportato nella figura a lato. In questo caso è visualizzata la configurazione di default ***115200, e, 8, 1***.

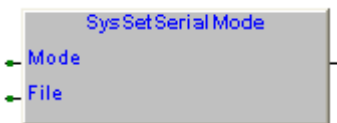
Symbol	Value	Type
Sm	-	SYSSERIALMODE
Baudrate	115200	UDINT
Parity	'E'	STRING
DataBits	8	USINT
StopBits	1	USINT
DTRManagement	2	USINT
DTRComplement	FALSE	BOOL
EchoFlush	FALSE	BOOL
DTROnTime	0	UINT
DTROffTime	0	UINT

7.9.2 SysSetSerialMode, set serial mode

Type	Library
Function	XTarget_07_0

SysSetSerialMode
Mode
File

Questa funzione imposta il modo di comunicazione definito nella struttura ***SYSSERIALMODE*** sulla porta seriale connessa al parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

Nel parametro ***Mode*** occorre definire l'indirizzo della struttura ***SYSSERIALMODE*** in cui è definito il modo seriale da impostare sulla porta. La funzione ritorna ***FALSE*** in caso di errore.

Parametri funzione:

Mode (@SYSSERIALMODE) Indirizzo struttura ***SYSSERIALMODE*** con il modo da impostare.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(BOOL) ***FALSE:*** Errore esecuzione. ***TRUE:*** Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9994010 Valore di ***File*** non definito.

9994020 Indirizzo struttura ***SYSSERIALMODE*** non corretto, verificare ***Mode***.

9994050 Errore valore di baud rate.

9994051 Errore valore di parità.

9994052 Errore valore bit di dato.

9994053 Errore valore bit di stop.

9994100 Errore esecuzione funzione.

9994990 Non implementata nel simulatore.

Esempi

Su fronte attivazione ingresso ***Di00M00*** viene salvato il modo impostato sulla porta seriale ***COM0*** nella variabile ***Sm***. Poi vengono modificati alcuni dati e poi impostata la porta seriale. Viene attivata l'uscita ***Do00M00***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Sm	SYSSERIALMODE	Auto	No	0	..	Serial mode data struct
3	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag

Esempio ST *(PTP116A300, ST_SysSetSerialMode)*

```
(* On input raising edge the serial mode is changed. *)

IF (Di00M00 <> Pulse) THEN
    Pulse:=Di00M00; (* Pulse flag *)

    IF (Di00M00) THEN
        IF (Fp = NULL) THEN Fp:=Sysfopen('COM0', 'rw'); END_IF;
        Do00M00:=SysGetSerialMode(ADR(Sm), Fp);
        Sm.Baudrate:=19200;
        Sm.Parity:='N';
        Sm.DTRManagement:=DTR_AUTO_WO_TIMES;
        Do01M00:=SysSetSerialMode(ADR(Sm), Fp);
    END_IF;
END_IF;
```

7.9.3 SetSMode, Set serial mode

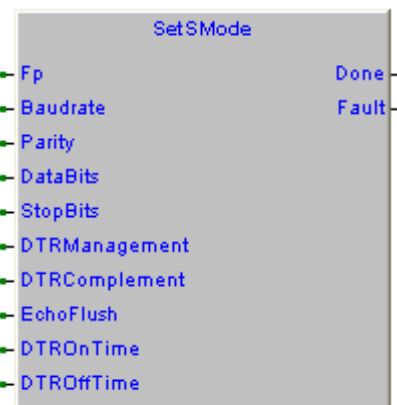
Type	Library
FB	ePLCUtyLib_C000

SetSMode
Fp — Done
Baudrate — Fault
Parity
DataBits
StopBits
DTRManagement
DTRComplement
EchoFlush
DTROnTime
DTROffTime

Questo blocco funzione esegue impostazione parametri di comunicazione porta seriale. Occorre passare al blocco funzione il puntatore al file di porta seriale su cui eseguire l'impostazione dei parametri di comunicazione cosi come viene ritornato dalla funzione di apertura file ***Sysfopen***. Per ulteriori chiarimenti sui vari parametri fare riferimento alla definizione modo comunicazione porta seriale ***SYSSERIALMODE***.

In caso di errore esecuzione (Esempio definizione di un parametro fuori range) il valore di impostazione della porta seriale non viene modificato e si attiverà l'uscita ***Fault***.

Attivando il modo di funzionamento ***DTR_AUTO_W_TIMES*** su ***DTRManagement*** è possibile con il parametro ***DTROnTime***, definire un tempo di attesa dopo attivazione segnale DTR prima della trasmissione dei dati su seriale. Con il parametro ***DTROffTime*** è possibile definire un tempo di attesa dopo la trasmissione dell'ultimo dato prima della disattivazione del segnale DTR.

Queste temporizzazioni sula gestione del segnale DTR permettono di gestire in modo automatico apparecchiature radiomodem, su cui deve essere attivato il comando di trasmissione prima di eseguire l'invio dei dati.

Fp (FILEP) Puntatore al file di porta seriale come ritornato da funzione ***Sysfopen***.

Baudrate (UDINT) Valore di baud rate porta seriale (da 300 a 115200 baud)

Parity (STRING[1]) Tipo di parità, valori possibili “E” pari, “O” dispari, “N” nessuna.

DataBits (USINT) Numero di bit frame dato, valori possibili 7, 8.

StopBits (USINT) Numero di bit di stop, valori possibili 1, 2.

DTRManagement (USINT) Modo di gestione del segnale DTR sulla porta seriale, vedi definizione.

DTRComplement (BOOL) ***FALSE:*** DTR normale, ***TRUE:*** DTR complementato.

EchoFlush (BOOL) ***FALSE:*** I dati trasmessi sono ritornati in ricezione. ***TRUE:*** I dati trasmessi sono ignorati.

DTROnTime (UINT) Tempo di attesa dopo attivazione segnale DTR prima di trasmissione caratteri (mS).

DTROffTime (UINT) Tempo di attesa dopo trasmissione ultimo dato prima e disattivazione segnale DTR (mS).

Done (BOOL) Impostazione modo eseguito correttamente.

Fault (BOOL) Errore su esecuzione blocco funzione

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10006010 Lettura modo corrente in errore la funzione ***SysGetSerialMode*** è ritornata con errore probabilmente non è stato indicato correttamente il file pointer ***Fp***.

10006020 Impostazione modo in errore la funzione ***SysSetSerialMode*** è ritornata con errore probabilmente uno dei parametri definiti non è nel range corretto.

Esempi

Al primo loop di esecuzione programma viene aperta la porta seriale ***COM0***, viene impostato il modo di comunicazione 19200, e, 8, 1. Ogni secondo viene inviato sulla porta seriale il carattere 'A' codice ascii 16#41.

Definizione variabili

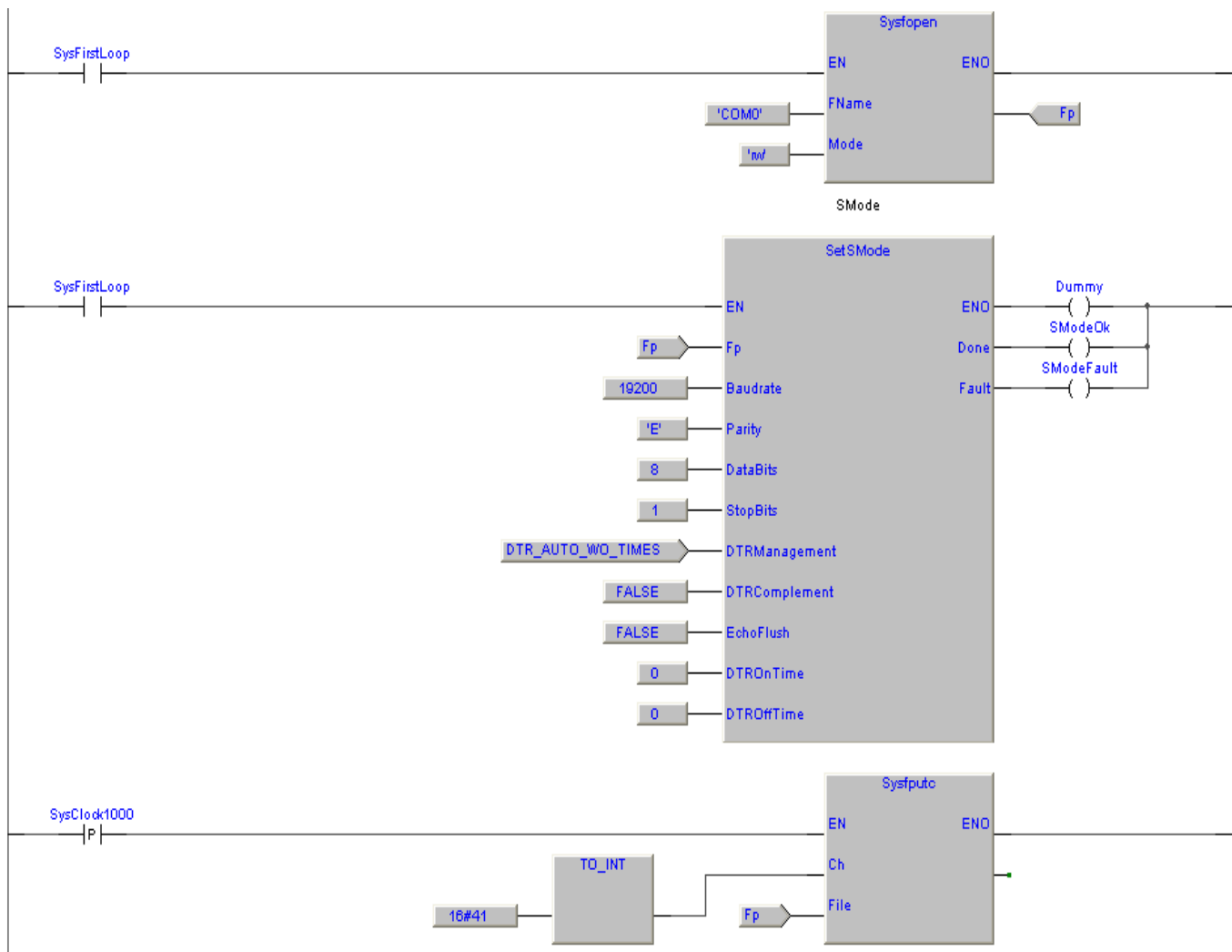
	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
3	SMode	SetSMode	Auto	No	0	..	Set serial mode
4	SModeOk	BOOL	Auto	No	FALSE	..	Set serial mode ok
5	SModeFault	BOOL	Auto	No	FALSE	..	Set serial mode fault

Esempio LD *(PTP119A000, SendCharacter)*

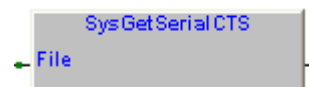
SysFirstLoop — Sysfopen: EN, ENO; 'COM0' → FName; 'rw' → Mode; → Fp

SMode

SysFirstLoop — SetSMode: EN → ENO (Dummy); Fp → Fp; Done (SModeOk); 19200 → Baudrate; Fault (SModeFault); 'E' → Parity; 8 → DataBits; 1 → StopBits; DTR_AUTO_WO_TIMES → DTRManagement; FALSE → DTRComplement; FALSE → EchoFlush; 0 → DTROnTime; 0 → DTROffTime

SysClock1000 (P) — Sysfputc: EN, ENO; 16#41 → TO_INT → Ch; Fp → File

7.9.4 SysGetSerialCTS, get serial CTS signal status

Type	Library
Function	XTarget_07_0

Questa funzione ritorna lo stato del segnale ***CTS*** della porta seriale connessa al parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

Parametri funzione:

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(BOOL) ***FALSE:*** Segnale CTS non attivo. ***TRUE:*** Segnale CTS attivo.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9993010 Valore di ***File*** non definito.

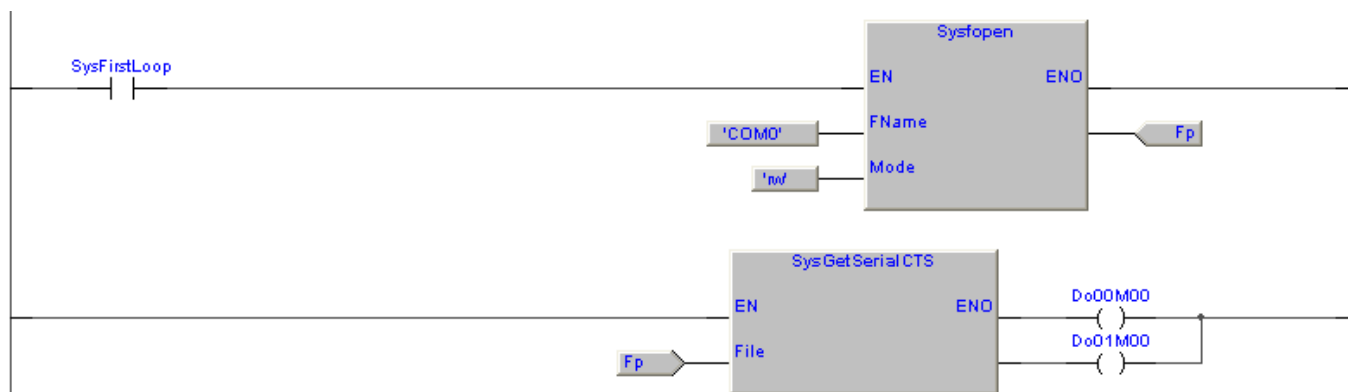
9993990 Non implementata nel simulatore.

Esempi

Lo stato del segnale ***CTS*** della porta seriale ***COM0*** è appoggiato sull'uscita ***Do01M00***.

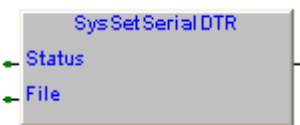
Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer

Esempio LD

7.9.5 SysSetSerialDTR, set DTR signal status

Type	Library
Function	XTarget_07_0

Questa funzione imposta lo stato del segnale ***DTR*** della porta seriale connessa al parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

Per poter gestire il segnale DTR occorre avere definito sulla porta seriale il valore ***DTR_OFF*** nella variabile ***DTRManagement*** nella struttura ***SYSSERIALMODE***.

Parametri funzione:

Status (BOOL) Stato segnale DTR su porta seriale

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

La funzione ritorna:

(BOOL) ***FALSE:*** Errore esecuzione. ***TRUE:*** Ok esecuzione.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9992010 Valore di ***File*** non definito.

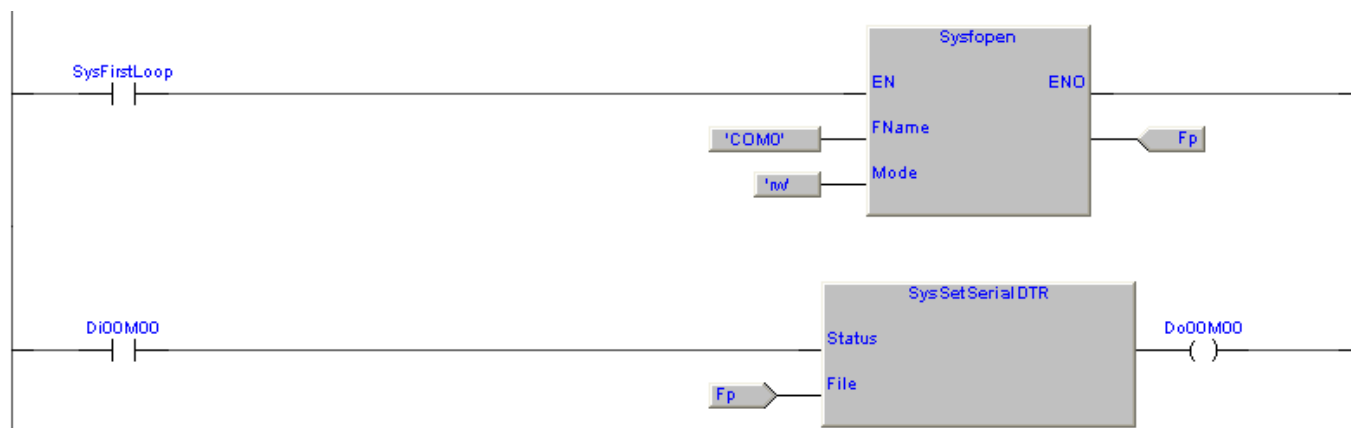
9992990 Non implementata nel simulatore.

Esempi

Lo stato dell'ingresso ***Di00M00*** viene trasferito sul segnale DTR della porta seriale ***COM0***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer

Esempio LD

7.10 Funzioni ed FB per gestione CAN bus

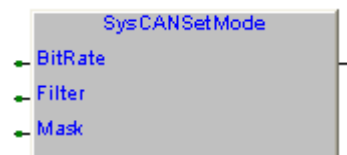
Il Controller Area Network, noto anche come CAN-bus, è uno standard seriale per bus di campo di tipo multicast, per collegare diverse unità di controllo elettronico (ECU). Il CAN è stato espressamente progettato per funzionare senza problemi anche in ambienti fortemente disturbati dalla presenza di onde elettromagnetiche e può utilizzare come mezzo trasmissivo una linea a differenza di potenziale bilanciata come la RS-485. L'immunità ai disturbi EMC può essere ulteriormente aumentata utilizzando cavi di tipo twisted pair (doppino intrecciato).

Sebbene inizialmente applicata in ambito automotive, come bus per autoveicoli, attualmente è usata in molte applicazioni industriali di tipo embedded, dove è richiesto un alto livello di immunità ai disturbi. Il bit rate può raggiungere 1 Mbit/s per reti lunghe meno di 40 m. Velocità inferiori consentono di raggiungere distanze maggiori (ad es. 125 kbit/s per 500 m). Il protocollo di comunicazione del CAN è standardizzato come ISO 11898-1 (2003).

7.10.1 SysCANSetMode, set the CAN controller mode

Type	Library
Function	XTarget_07_0

Questa funzione imposta il modo sul controller CAN. E' possibile definire il valore di bit rate, il filtro e la maschera di accettazione pacchetti.

La funzione ritorna TRUE se eseguita correttamente, FALSE se errore.

Parametri funzione:

BitRate (USINT) Valore definizione bit rate CAN bus, ***CAN bit rate definition***.

Filter (UDINT) Valore di filtro pacchetti CAN.

Mask (UDINT) Valore di maschera filtro pacchetti CAN.

La funzione ritorna:

(BOOL) ***TRUE:*** Eseguita correttamente.
FALSE: In caso di errore esecuzione, esempio parametri errati.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9957005 Funzione non supportata.

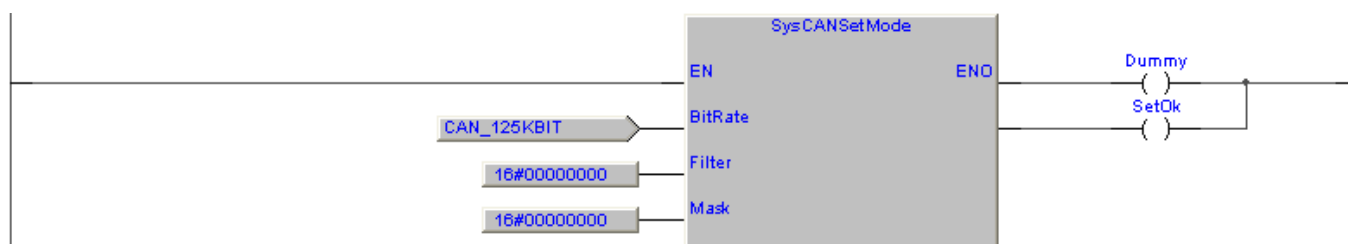
9957990 Non implementata nel simulatore.

Esempi

E' riportato un semplice programma che esegue l'impostazione del controller CAN con bit rate a 125 Kbit. Tutti i pacchetti in arrivo sono ricevuti dal controller.

Definizione variabili

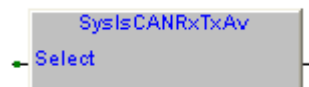
	Name	Type	Address	Array	Init value	Attribute	Description
1	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
2	SetOk	BOOL	Auto	No	FALSE	..	CAN set mode Ok

Esempio LD

7.10.2 SysIsCANRxTxAv, checks if CAN Rx or Tx is available

Type	Library
Function	XTarget_07_0

Questa funzione controlla:

Select:=FALSE: Se vi è almeno un messaggio nel buffer di ricezione CAN.
Select:=TRUE: Se vi è spazio per un messaggio nel buffer di trasmissione CAN.

SysIsCANRxTxAv
Select

La funzione ritorna TRUE se la condizione selezionata è vera.

Parametri funzione:

Select (BOOL) ***FALSE:*** Se vi è almeno un messaggio nel buffer di ricezione CAN.
TRUE: Se vi è spazio per un messaggio nel buffer di trasmissione CAN.

La funzione ritorna:

(BOOL) ***TRUE:*** Condizione selezionata è vera.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9956005 Funzione non supportata.

9956990 Non implementata nel simulatore.

Esempi

E' riportato un semplice programma che esegue il controllo se un messaggio CAN è stato ricevuto, ne esegue la ricezione ed invia in uscita sulla porta seriale ***COM0*** la struttura del messaggio ricevuto.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	CANMsg	SYSCANMESSAGE	Auto	No	0	..	CAN message
3	NrOfChars	INT	Auto	No	0	..	Number of printed chars

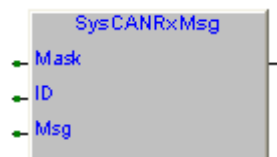
Esempio ST

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

(* Here check if a CAN message is available and receive it. *)

    IF (SysIsCANRxTxAv(FALSE)) THEN
        IF (SysCANRxMsg(16#00000000, 16#00000000, ADR(CANMsg))) THEN

            NrOfChars:=SysVarfprintf(Fp, 'Length:%04d$r$n', USINT_TYPE, ADR(CANMsg.Length));
            NrOfChars:=SysVarfprintf(Fp, 'MsgID:%04d$r$n', UDINT_TYPE, ADR(CANMsg.MsgID));
            NrOfChars:=SysVarfprintf(Fp, 'Data[0]:%02X$r$n', UDINT_TYPE, ADR(CANMsg.Data[0]));

        END_IF;
    END_IF;
```

7.10.3 SysCANRxMsg, receives a CAN message

Type	Library
Function	XTarget_07_0

Questa funzione riceve un messaggio CAN e lo trasferisce nella variabile il cui indirizzo è definito in ***Msg***. E' possibile definire un ***Mask*** ed un ***ID*** per ricevere i soli messaggi CAN desiderati.

La funzione ricerca nello stack dei messaggi un messaggio il cui ID posto in AND con ***Mask*** coincide con ***ID*** messo in AND con ***Mask***. La funzione ritorna TRUE se messaggio ricevuto.

Parametri funzione:

Mask (UDINT) Codice maschera ID messaggio.

ID (UDINT) ID check ID messaggio.

Msg (@SYSCANMESSAGE) Indirizzo buffer messaggio ricevuto.

La funzione ritorna:

(BOOL) ***TRUE:*** Messaggio ricevuto.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9955005 Funzione non supportata.

9955990 Non implementata nel simulatore.

Esempi

E' riportato un semplice programma che esegue la ricezione di qualsiasi messaggio CAN ed esegue l'invio in uscita sulla porta seriale ***COM0*** della struttura del messaggio ricevuto.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	CANMsg	SYSCANMESSAGE	Auto	No	0	..	CAN message
3	NrOfChars	INT	Auto	No	0	..	Number of printed chars

Esempio ST

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

(* Here receive a CAN message. *) *)

    IF (SysCANRxMsg(16#3FFFFFFF, 16#00000000, ADR(CANMsg))) THEN

        NrOfChars:=SysVarfprintf(Fp, 'Length:%04d$r$n', USINT_TYPE, ADR(CANMsg.Length));
        NrOfChars:=SysVarfprintf(Fp, 'MsgID:%04d$r$n', UDINT_TYPE, ADR(CANMsg.MsgID));
        NrOfChars:=SysVarfprintf(Fp, 'Data[0]:%02X$r$n', UDINT_TYPE, ADR(CANMsg.Data[0]));

    END_IF;
```

7.10.4 SysCANTxMsg, transmit a CAN message

Type	Library
Function	XTarget_07_0

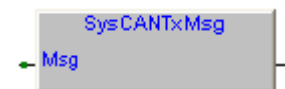
Questa funzione trasmette un messaggio CAN, occorre creare il messaggio e poi passarne alla funzione l'indirizzo in ***Msg***.

La funzione ritorna TRUE se messaggio trasmesso.

Parametri funzione:

Msg (@SYSCANMESSAGE) Indirizzo buffer messaggio da trasmettere.

La funzione ritorna:

(BOOL) ***TRUE:*** Messaggio trasmesso.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9954005 Funzione non supportata.

9954990 Non implementata nel simulatore.

Esempi

E' riportato un semplice programma che esegue la trasmissione di un messaggio CAN.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	CANMsg	SYSCANMESSAGE	Auto	No	0	..	CAN message
2	TxOk	BOOL	Auto	No	FALSE	..	Transmission Ok

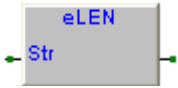
Esempio ST

```
(* Here check if there is a space in Tx buffer and send a CAN message. *)

    IF (SysIsCANRxTxAv(TRUE)) THEN
        CANMsg.RmReq:=FALSE; (* eFALSE:Data frame, eTRUE:Remote request *)
        CANMsg.Length:=2; (* Data length *)
        CANMsg.MsgID:=16#00000000; (* Message ID (FF:Bit 31) (11 or 29 Bit) *)
        CANMsg.Data[0]:=16#01; (* Message data *)
        CANMsg.Data[1]:=16#00; (* Message data *)
        TxOk:=SysCANTxMsg(ADR(CANMsg)); (* Transmission Ok *)
    END_IF;
```

7.11 Funzioni ed FB per gestione stringhe

7.11.1 eLEN, string length

Type	Library
Function	ePLCStdLib_B000

Questa funzione ritorna la lunghezza (Espressa in numero di caratteri) della stringa definita in ***Str***.

Parametri funzione:

Str (@USINT) Pointer alla stringa di cui calcolare lunghezza.

La funzione ritorna:

(INT) Numero di caratteri della stringa.

Esempi

Viene calcolata la lunghezza della stringa ***'Hello!'*** ed il numero di caratteri che compongono la stringa viene trasferito nella variabile ***Length***. Il risultato del calcolo è ***6***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Length	INT	Auto	No	0	..	String length

Esempio LD

7.11.2 eFIND, string find

Type	Library
Function	ePLCStdLib_B000

Questa funzione cerca la posizione del carattere di inizio della prima apparizione di ***In2*** in ***In1***. Se nessuna apparizione viene trovata, la funzione ritorna ***0***.

Se la stringa ***In2*** è trovata nella stringa ***In1***, viene ritornata la posizione dove si trova.

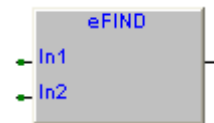
Esempio: ***eFIND(In1:='abcd', In2:='bc')***. Ha come risultato ***2***.

Parametri funzione:

In1 (@USINT) Pointer alla stringa dove effettuare la ricerca.

In1 (@USINT) Pointer alla stringa da ricercare.

La funzione ritorna:

(INT) Posizione dove la stringa ***In2*** è stata trovata. ***0*** se stringa non trovata.

Esempi

Viene eseguita la ricerca della stringa ***'lo'*** nella stringa ***'Hello world!'***. La posizione trovata è 4 e viene trasferita nella variabile ***Position***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Position	INT	Auto	No	0	..	StrToFind position
2	StrSource	STRING	Auto	[32]		..	String were looking
3	StrToFind	STRING	Auto	[32]		..	String to find

Esempio ST

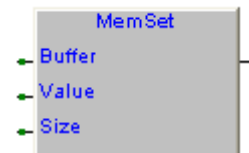
```
(* Find the position where is StrToFind in StrSource. *)

    StrSource:='Hello world!';
    StrToFind:='lo';

    Position:=eFIND(ADR(StrSource), ADR(StrToFind));
```

7.11.3 MemSet, memory set

Type	Library
Function	ePLCUtyLib_C000

Questa funzione trasferisce in ***Buffer*** il valore definito in ***Value*** per il numero di bytes definito in ***Size***.

La funzione ritorna il numero definito in ***Size***.

MemSet
Buffer
Value
Size

Parametri funzione:

Buffer (@USINT) Pointer al buffer di memoria dove trasferire ***Value***.

Value (USINT) Valore da trasferire nel buffer di memoria.

Size (UDINT) Numero di volte in cui ***Value*** è trasferito in ***Buffer***.

La funzione ritorna:

(UDINT) Valore definito in ***Size***.

Esempi

Viene eseguito l'azzeramento di tutti i bytes della stringa ***StrBuffer*** scrivendo il valore ***0*** in tutta la sua lunghezza.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	StrBuffer	STRING	Auto	[32]		..	String buffer
2	RetValue	UDINT	Auto	No	0	..	Return value

Esempio ST

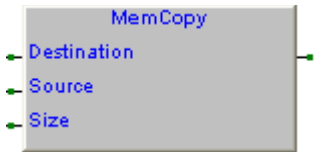
```
(* The 'StrBuffer' variable is set to '0'. *)

    RetValue:=MemSet(ADR(StrBuffer), 0, 32); (* Return value *)
```

7.11.4 MemCopy, memory copy

Type	Library
Function	ePLCUtyLib_C000

Questa funzione copia al massimo il numero di bytes definito in ***Size*** a partire dall'area di memoria a cui punta ***Source***, verso l'area che inizia da ***Destination***.

La funzione ritorna il numero definito in ***Size***.

MemCopy
Destination
Source
Size

Parametri funzione:

Destination (@USINT) Pointer al buffer di memoria di destinazione.

Source (@USINT) Pointer al buffer memoria sorgente.

Size (UDINT) Numero di bytes da trasferire.

La funzione ritorna:

(UDINT) Valore definito in ***Size***.

Esempi

Viene eseguita la copia della stringa ***SString*** nella stringa ***DString***. Al termine la stringa risultante sarà “HelloHello”.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RetValue	UDINT	Auto	No	0	..	Return value
2	DString	STRING	Auto	[32]		..	Destination string
3	SString	STRING	Auto	[8]	Hello	..	Source string

Esempio ST

```
(* The 'SString' is copied twice into 'DString'. *)

    RetValue:=MemCopy(ADR(DString), ADR(SString), SIZEOF(SString)); (* Return value *)
    RetValue:=MemCopy(ADR(DString)+LEN(DString), ADR(SString), SIZEOF(SString)); (* Return value *)
```

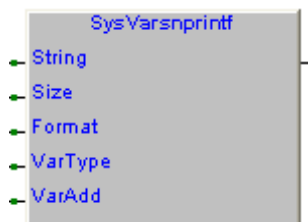
7.11.5 SysVarsnprintf, variable print to string

Type	Library
Function	XTarget_08_0

Questa funzione trasferisce in ***String*** la stampa formattata di una variabile. Il valore stampato ritornato nella variabile stringa non può superare la lunghezza definita in ***Size***.

La stringa ***Format*** specifica il formato con il quale stampare la variabile. Mentre in ***VarType*** è indicato il tipo di variabile ed in ***VarAdd*** il suo indirizzo.

La funzione ritorna il numero di caratteri trasferiti nella variabile ***String***. EOF in caso di errore.

SysVarsnprintf
String
Size
Format
VarType
VarAdd

Parametri funzione:

String (@USINT) Pointer all'array dove deve essere trasferito il risultato della stampa.

Size (UINT) Numero di caratteri da trasferire nella variabile ***String***. Il numero definito è comprensivo del codice di fine stringa '\0'. Se la lunghezza della stringa di output supera il limite di ***Size*** byte, viene troncata al numero di byte indicato.

Format (STRING[80]) Ha due tipi di argomenti, i caratteri ordinari che vengono copiati nella variabile ***String*** di uscita, e le specifiche di conversione, contraddistinte dal simbolo percentuale (%) e da un carattere che specifica il formato con il quale stampare la variabile definita.

VarType (USINT) Tipo variabile, come indicato nella tabella ***Variable types definition***.

VarAdd (UDINT) Indirizzo variabile.

La funzione ritorna:

(INT) Numero di caratteri comprensivo codice fine stringa '\0' trasferiti in variabile ***String***.
EOF: Errore esecuzione.

Codici di errore

In caso di errore la funzione torna con ***EOF*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9997100 Tipo variabile non gestito, controllare ***VarType***.

9997200 Il valore di ***Size*** limita la formattazione della stringa in uscita.

Esempi

Su fronte attivazione ingresso ***Di00M00*** viene incrementata la variabile ***Counter*** e la stampa del suo valore trasferita nell'array ***StringOut***. Il valore presente in ***StringOut*** viene poi inviato sulla porta seriale ***COM0***. Nella variabile ***NrOfChars*** viene caricato il numero di caratteri trasferiti in ***StringOut*** ed inviati in uscita sulla porta seriale.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Port COM0 file pointer
2	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag
3	Counter	UDINT	Auto	No	0	..	Counter
4	NrOfChars	INT	Auto	No	0	..	Number of printed chars
5	StringOut	USINT	Auto	[0..31]	32(0)	..	String output
6	i	INT	Auto	No	0	..	Auxiliary counter
7	Ch	INT	Auto	No	0	..	Character written

Esempio ST *(PTP116A100, ST_SysVarsnprintf)*

```
(* Here the COM0 port is opened in read/write. *)

    IF (Fp = NULL) THEN
        Fp:=Sysfopen('COM0', 'rw'); (* Port COM0 file pointer *)
    END_IF;

(* Check if the COM0 port is open. *)

    IF (Fp <> NULL) THEN
```

```
        (* Check if input is activated. *)

        IF (Di00M00 <> Pulse) THEN
            Pulse:=Di00M00; (* Pulse flag *)

            (* On input raising edge the counter value is printed. *)

            IF (Di00M00) THEN
                Counter:=Counter+1; (* Counter *)
                NrOfChars:=SysVarsnprintf(ADR(StringOut), 32, 'Counter:%04d$r$n', UDINT_TYPE, ADR(Counter));

                (* Copy the printed result to serial port. *)

                FOR i:=0 TO NrOfChars DO Ch:=Sysfputc(TO_INT(StringOut[i]), Fp); END_FOR;
            END_IF;
        END_IF;
    END_IF;
```

In questo esempio viene eseguito il merge tra le stampe del valore di due variabili. Eseguire il merge può essere molto utile per avere un unica stringa contenente la stampa del valore di piu variabili.

Mettendo in debug la variabile ***Result*** vedremo la stringa ***Var[0]:12 Var[1]:34***. Avendo bloccato la stampa a 12 caratteri il valore di ***Var[0]*** sarà stampato correttamente fino ad un massimo di 4 cifre (7 caratteri stringa, 4 caratteri valore più terminatore stringa '\0'). Per valori di ***Var[0]*** superiori a 9999 non saranno più stampate le cifre meno significative.

Il valore di ***Var[1]*** sarà stampato immediatamente dopo il valore di ***Var[0]***. Da notare l'offset decrementato di 1 per sovrascrivere il terminatore stringa '\0'. Per garantire la stampa di un massimo di 4 anche per ***Var[1]*** è stato definito un ***Size*** di 13, la stringa di testo inizia infatti con un carattere di space per separarla dalla stampa del valore della variabile precedente.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	NrOfChars	INT	Auto	No	0	..	Number of printed chars
2	Var	UDINT	Auto	[0..1]	12,34	..	Variables
3	Result	STRING	Auto	[32]		..	

Esempio ST

```
(* -------------------------------------------------------------------------- *)
(* EXECUTE A VARIABLES PRINT MERGE                                            *)
(* -------------------------------------------------------------------------- *)
(* Print the variable values, merging them into a single string. *)

    NrOfChars:=SysVarsnprintf(ADR(Result), 12, 'Var[0]:%d', UDINT_TYPE, ADR(Var[0]));
    NrOfChars:=SysVarsnprintf(ADR(Result[LEN(Result)]), 13, ' Var[1]:%d', UDINT_TYPE, ADR(Var[1]));

(* [End of file] *)
```

7.11.6 SysVarsscanf, extracts values from string

Type	Library
Function	XTarget_08_0

Questa funzione legge la stringa ***String*** e ne interpreta il contenuto basandosi sul parametro ***Format***.

La stringa ***Format*** specifica il formato con il quale interpretare la variabile, in ***VarType*** è indicato il tipo di variabile ed in ***VarAdd*** il suo indirizzo.

La funzione ritorna TRUE se valore variabile trovato, in caso contrario FALSE.

Parametri funzione:

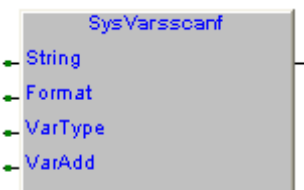
String (@USINT) Pointer alla stringa da leggere.

Format (STRING[80]) Ha due tipi di argomenti, i caratteri ordinari che vengono controllati nella variabile ***String*** da leggere e le specifiche di conversione, contraddistinte dal simbolo percentuale (%) e da un carattere che specifica il formato con il quale stampare la variabile definita.

VarType (USINT) Tipo variabile, come indicato nella tabella ***Variable types definition***.

VarAdd (UDINT) Indirizzo variabile.

La funzione ritorna:

(BOOL) ***TRUE:*** Valore variabile acquisito.

Codici di errore

In caso di errore la funzione torna con ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9999100 Tipo variabile non gestito, controllare ***VarType***.

Esempi

E' riportato un programma che esegue la lettura di una stringa ***InputString*** valorizzata con il testo ***Value:123***. Su fronte attivazione ingresso ***Di00M00*** sono eseguite tre diverse funzioni ***SysVarsscanf*** tutte sulla stringa ***InputString*** ma con diverse definizioni di ***Format***. Le prime due hanno esito positivo e le variabili ***Variable[0]*** e ***Variable[1]*** saranno valorizzate con il valore ***123***. La terza avrà esito negativo, la variabile ***Variable[2]*** sarà azzerata.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Pulse	BOOL	Auto	No	FALSE	..	Pulse flag
2	Variable	UDINT	Auto	[0..2]	3(0)	..	Variable
3	InputString	STRING	Auto	[32]	Value:123	..	Input string
4	Result	BOOL	Auto	[0..2]	3(0)	..	Function result

Esempio ST *(PTP116A200, ST_SysVarsscanf)*

```
(* Check if input is activated. *)

    IF (Di00M00 <> Pulse) THEN
        Pulse:=Di00M00; (* Pulse flag *)

        IF (Di00M00) THEN
            Result[0]:=SysVarsscanf(ADR(InputString), 'Value:%d', UDINT_TYPE, ADR(Variable[0]));
            Result[1]:=SysVarsscanf(ADR(InputString)+6, '%d', UDINT_TYPE, ADR(Variable[1]));
            Result[2]:=SysVarsscanf(ADR(InputString), '%d', UDINT_TYPE, ADR(Variable[2]));
        END_IF;
    END_IF;
```

7.12 Funzioni ed FB per gestione moduli periferici

7.12.1 SysPhrInfos, get infos from peripheral modules

Type	Library
Funzione	XTarget_07_0

SysPhrInfos
Module
InfoID
Info

Questa funzione esegue l'acquisizione di informazioni dai moduli periferici. Viene trasferita nella variabile stringa il cui indirizzo è passato in ***Info***, l'informazione indicata da ***InfoID*** relativa al modulo indicato in ***Module***.

La funzione ritorna ***TRUE*** se correttamente eseguita, in caso contrario ***FALSE***.

Parametri funzione:

Module (USINT) Occorre specificare l'indirizzo di modulo da cui eseguire l'acquisizione (Range da 0 a 15). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito.

InfoID (USINT) Occorre specificare l'ID della informazione richiesta.

Value	Description
0	Ritorna codice prodotto
1	Ritorna codice programma

Info (STRING[10]) Indirizzo variabile dove trasferire l'informazione.

La funzione ritorna:

(BOOL) ***FALSE:*** Errore esecuzione. ***TRUE:*** Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna con ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9990100 Il modulo indirizzato in ***Module*** non è presente.

9990110 Il valore di ***InfoID*** non è corretto.

9990200 L'informazione richiesta non è supportata dal modulo.

9990210 Errore durante la richiesta della informazione dal modulo.

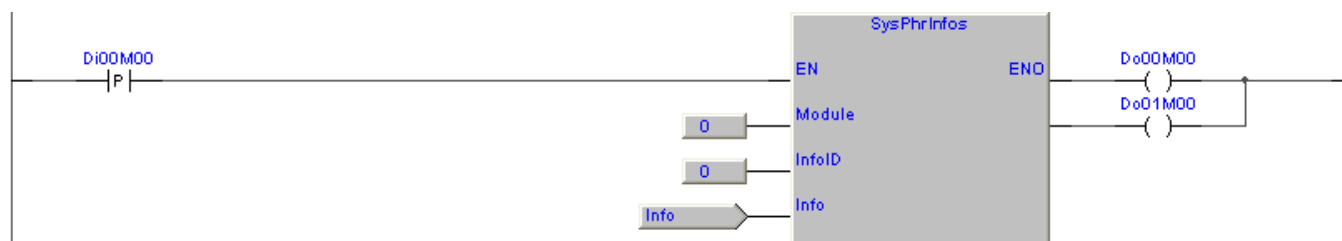
9990990 Non implementata nel simulatore.

Esempi

E' riportato un programma che esegue la lettura del codice prodotto dal modulo di estensione con indirizzo 0. Il codice ritornato è trasferito nella variabile ***Info***. Il ritorno della funzione è trasferito sull'uscita ***Do01M00***.

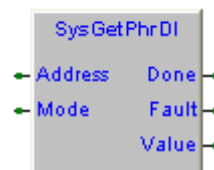
Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Info	STRING	Auto	[10]		..	Info returned from module

Esempio LD

7.12.2 SysGetPhrDI, get peripheral digital input

Type	Library
FB	XTarget_07_0

Questo blocco funzione esegue l'acquisizione degli ingressi digitali dai moduli periferici. Il blocco funzione ritorna lo stato degli ingressi digitali dal modulo indicato in ***Address*** in base al comando di ***Mode*** definito.

Per acquisire gli ingressi digitali presenti sul modulo CPU occorre definire ***Address*** 255 e ***Mode*** DI_8_LL.

Address (USINT) Occorre specificare l'indirizzo di modulo da cui eseguire l'acquisizione degli ingressi digitali (Range da 0 a 255). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito. L'indirizzo 255 indica il modulo CPU.

Mode (USINT) Occorre specificare il modo di acquisizione ingressi digitali, ***Digital input mode***.

Done (BOOL) Dato acquisito, viene attivato se acquisizione ingressi digitali terminata.

Fault (BOOL) Errore di acquisizione, viene attivato in caso di errore nella sequenza di acquisizione.

Value (UDINT) Ritorna lo stato degli ingressi digitali acquisiti.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

9985050 Errore allocazione blocco funzione.

9985070 Errore versione blocco funzione.

9985100 Il modulo indirizzato in ***Address*** non è presente.

9985110~24 Il modo acquisizione definito in ***Mode*** non è corretto.

9985200~13 Errore durante l'esecuzione della lettura ingressi dal modulo periferico.

Esempi

Viene acquisito lo stato degli 8 ingressi bassi (Da Inp 0 a Inp 7) del modulo di con indirizzo 0. Se dato valido viene attivata la variabile ***FBDone*** se errore di acquisizione viene attivata la variabile ***FBFault***. Il valore acquisito nel range da 0x00 a 0xFF è trasferito nella variabile ***InpValue***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	SysGetPhrDI	Auto	No	0	..	FB SysGetPhrDI data
2	InpValue	UDINT	Auto	No	0	..	Digital input value
3	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
4	FBDone	BOOL	Auto	No	FALSE	..	FB done
5	FBFault	BOOL	Auto	No	FALSE	..	FB fault

Esempio LD *(PTP116A100, LD_SysGetPhrDI)*

Esempio IL *(PTP116A100, IL_SysGetPhrDI)*

```
(* Read Inp 0 to 7 from module with address 0. *)

    LD 0
    ST FBData.Address

    LD DI_8_LL
    ST FBData.Mode

    CAL FBData

    LD FBData.Value
    ST InpValue
```

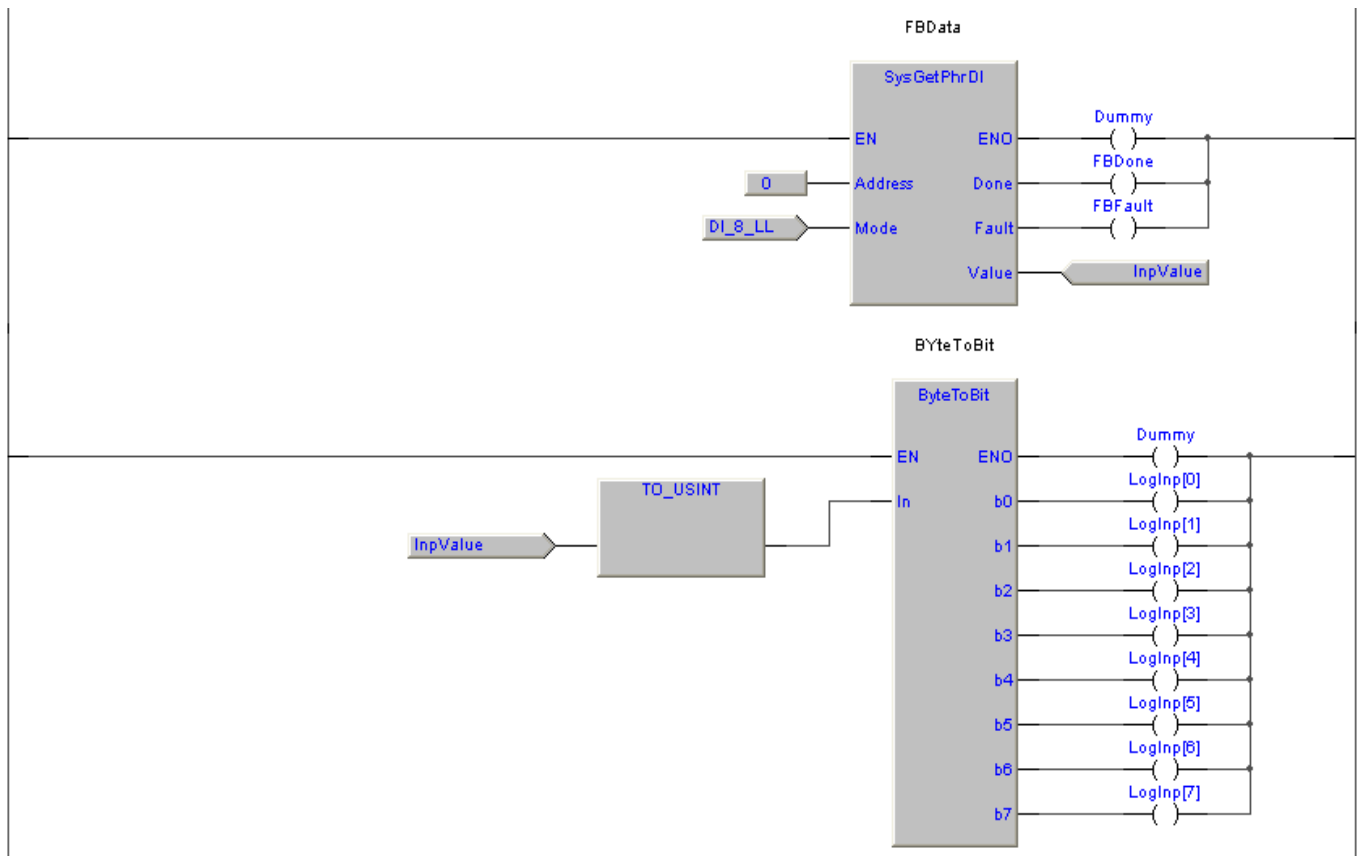
Esempio ST *(PTP116A100, ST_SysGetPhrDI)*

```
(* Read Inp 0 to 7 from module with address 0. *)

    FBData.Address:=0;
    FBData.Mode:=DI_8_LL;
    FBData(); (* Execute FB *)
    InpValue:=FBData.Value; (* Digital input value *)
```

Questo esempio è una evoluzione dell'esempio precedente. Utilizzando il blocco funzione ***ByteToBit*** lo stato degli 8 ingressi bassi (Da Inp 0 a Inp 7) del modulo con indirizzo 0, è appoggiato su di un array di BOOL.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	SysGetPhrDI	Auto	No	0	..	FB SysGetPhrDI data
2	InpValue	UDINT	Auto	No	0	..	Digital input value
3	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
4	FBDone	BOOL	Auto	No	FALSE	..	FB done
5	FBFault	BOOL	Auto	No	FALSE	..	FB fault
6	BYteToBit	ByteToBit	Auto	No	0	..	FB ByteToBit
7	LogInp	BOOL	Auto	[0..7]	8(0)	..	Logic inputs

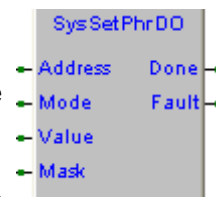
Esempio LD *(PTP119A000, LogicInputAcquisition)*

7.12.3 SysSetPhrDO, set peripheral digital output

Type	Library
FB	XTarget_11_0

Questo blocco funzione esegue impostazione delle uscite digitali sui moduli periferici indirizzati con ***Address*** ed in base al comando di ***Mode*** definito.

Per non perturbare la gestione delle uscite logiche eseguita dall'immagine di processo, in ***Mask*** è possibile definire su quali output si desidera operare. In pratica l'FB opererà solo sugli output il cui relativo bit di mask è attivo.

Per gestire le uscite digitali presenti sul modulo CPU occorre definire ***Address*** 255 e ***Mode*** DO_8_LL.

SysSetPhrDO
Address Done
Mode Fault
Value
Mask

Address (USINT) Occorre specificare l'indirizzo del modulo su cui eseguire l'impostazione delle uscite digitali (Range da 0 a 255). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito. L'indirizzo 255 indica il modulo CPU.

Mode (USINT) Occorre specificare il modo di gestione uscite digitali, ***Digital output mode***.

Value (UDINT) Impostare il valore da trasferire sulle uscite digitali.

Mask (UDINT) Maschera outputs da gestire.

Done (BOOL) Dato settato. Viene attivato per un loop al termine del settaggio delle uscite digitali.

Fault (BOOL) Errore. Viene attivato in caso di errore nella sequenza di settaggio.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

9984050 Errore allocazione blocco funzione.

9984060 Terminato spazio memoria rilocabile, non è possibile eseguire l''FB.

9984070 Errore versione blocco funzione.

9984100 Il modulo indirizzato in ***Address*** non è presente.

9984110~8 Il modo gestione definito in ***Mode*** non è corretto.

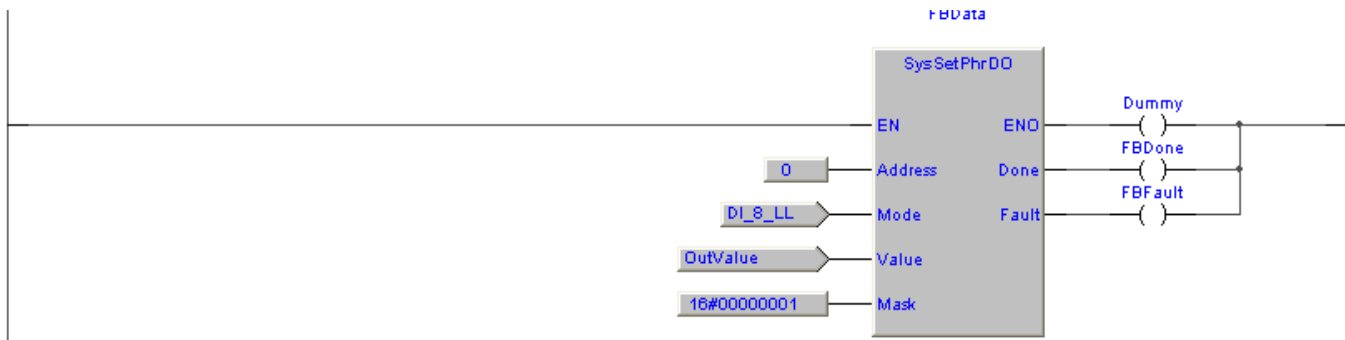
9984200~6 Errore durante l'esecuzione gestione uscite digitali sul modulo periferico.

Esempi

Viene trasferito il valore della variabile ***OutValue*** sulla uscita Out 0 del modulo con indirizzo 0.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	SysSetPhrDO	Auto	No	0	..	FB SysSetPhrDO data
2	OutValue	UDINT	Auto	No	0	..	Digital output value
3	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
4	FBDone	BOOL	Auto	No	FALSE	..	FB done
5	FBFault	BOOL	Auto	No	FALSE	..	FB fault

Esempio LD *(PTP116A400, LD_SysSetPhrDO)*

Esempio IL *(PTP116A400, IL_SysSetPhrDO)*

```
(* Manage digital outputs Out 0 to Out 7 on module with address 0. *)

    LD 0
    ST FBData.Address

    LD DO_8_LL
    ST FBData.Mode

    LD OutValue
    LD FBData.Value

    LD 16#00000001
    LD FBData.Mask

    CAL FBData
```

Esempio ST *(PTP116A400, ST_SysSetPhrDO)*

```
(* Manage digital outputs Out 0 to Out 7 on module with address 0. *)

    FBData.Address:=0;
    FBData.Mode:=DO_8_LL;
    FBData.Value:=OutValue; (* Digital ouput value *)
    FBData.Mask:=16#00000001; (* Output mask *)
    FBData(); (* Execute FB *)
```

7.12.4 SysGetAnInp, get analog input

Type	Library
FB	XTarget_07_0

Questo blocco funzione esegue l'acquisizione dell'ingresso analogico dal modulo di acquisizione. Il blocco funzione gestisce vari modi di acquisizione in funzione del modulo analogico a cui fa riferimento.

Per acquisire gli ingressi digitali presenti sul modulo CPU occorre definire ***Address*** 255 e ***Mode*** AD_VOLT_0_10_COMMON.

Address (USINT) Occorre specificare l'indirizzo di modulo da cui eseguire l'acquisizione degli ingressi analogici (Range da 0 a 255). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito. L'indirizzo 255 indica il modulo CPU.

Channel (USINT) Occorre specificare l'indirizzo del canale sul modulo (Range da 0x00 a 0x0F). Se viene settato un indirizzo di canale non presente, si interrompe l'esecuzione e viene settato il bit di ***Fault***.

Mode (USINT) Occorre specificare il modo di acquisizione analogica, ***Analog to digital mode***.

Done (BOOL) Dato analogico acquisito, viene attivato al termine della acquisizione analogica.

Fault (BOOL) Errore di acquisizione, viene attivato in caso di errore nella sequenza di acquisizione.

Value (REAL) Ritorna il valore di acquisizione espresso nella unità definita dal modo di acquisizione. Potrebbe essere un valore NaN (Not A Number) ad indicare un problema nell'acquisizione, tipicamente sensore rotto.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

9983050 Errore allocazione blocco funzione.

9983060 Terminato spazio memoria rilocabile, non è possibile eseguire l''FB.

9983070 Errore versione blocco funzione.

9983080 Impossibile inizializzare il modulo.

9983100 Il modulo indirizzato in ***Address*** non è presente.

9983110~1 Il modulo indirizzato non supporta i comandi acquisizione analogica.

9983150 Il valore ritornato dal modulo analogico non è corretto.

9983200 Il modo di acquisizione definito in ***Mode*** non è gestito dal modulo.

9983210 Errore nella acquisizione analogica dal modulo.

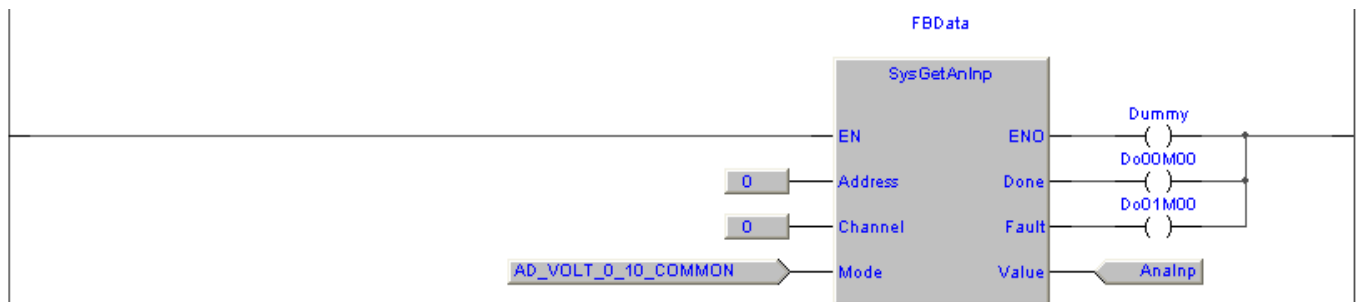
9983300 Il canale definito in ***Channel*** non è gestito dal modulo.

Esempi

Viene eseguita l'acquisizione analogica dal canale 0 del modulo 0 in modo 0-10 Volt. Se dato valido viene attivata l'uscita digitale ***Do00M00*** se errore di conversione viene attivata l'uscita digitale ***Do01M00***. Il dato analogico acquisito nel range da 0.000 a 9.999 è trasferito nella variabile ***AnaInp***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	SysGetAnInp	Auto	No	0	..	FB SysGetPhrDI data
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
3	AnaInp	REAL	Auto	No	0	..	Analog value (Volts)

Esempio LD *(PTP116A100, LD_SysGetAnInp)*

Esempio IL *(PTP116A100, IL_SysGetAnInp)*

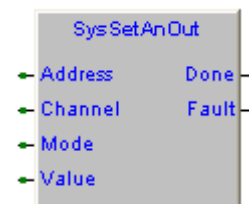
```
(* Acquires analog input 0 from module. *)

    LD   0
    ST   FBData.Address (* Set module address *)

    LD   0
    ST   FBData.Channel (* Set channel *)

    LD   AD_VOLT_0_10_COMMON
    ST   FBData.Mode (* Set acquisition mode *)

    CAL FBData (* Call the SysGetAnInp function block *)

    LD   FBData.Done
    ST   Do00M00 (* The output is active if data is acquired *)

    LD   FBData.Fault
    ST   Do01M00 (* The output is active if execution fault *)

    LD   FBData.Value
    ST   AnaInp (* Store the acquired value *)
```

Esempio ST *(PTP116A100, ST_SysGetAnInp)*

```
(* Acquires analog input 0 from module. *)

    FBData(Address:=0, Channel:=0, Mode:=AD_VOLT_0_10_COMMON); (* Call the SysGetAnInp FB *)

    Do00M00:=FBData .Done; (* The output is active if data is acquired *)
    Do01M00:=FBData .Fault; (* The output is active if execution fault *)
    VarReal:=FBData .Value; (* Store the acquired value *)
```

7.12.5 SysSetAnOut, set analog output

Type	Library
FB	XTarget_07_0

Questo blocco funzione esegue il set del valore sull'uscita analogica sul modulo di uscita. Il blocco funzione gestisce vari modi di uscita in funzione del modulo analogico a cui fa riferimento.

Address (USINT) Occorre specificare l'indirizzo del modulo su cui settare il valore analogico (Range da 0x00 a 0x0F). Il valore 0x00 indica il primo modulo di estensione, 0x01 il secondo e così di seguito.

Channel (USINT) Occorre specificare l'indirizzo del canale sul modulo (Range da 0x00 a 0x0F). Se viene settato un indirizzo di canale non presente, si interrompe l'esecuzione e viene settato il bit di ***Fault***.

Mode (USINT) Occorre specificare il modo di gestione uscita analogica, ***Digital to analog mode***.

Value (REAL) Occorre specificare il valore di uscita espresso nella unità definita dal modo di gestione.

Done (BOOL) Dato settato. Viene attivato per un loop al termine del settaggio dell'uscita analogica.

Fault (BOOL) Errore. Viene attivato in caso di errore nella sequenza di settaggio.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

9982050 Errore allocazione blocco funzione.

9982060 Terminato spazio memoria rilocabile, non è possibile eseguire l''FB.

9982070 Errore versione blocco funzione.

9982080 Impossibile inizializzare il modulo.

9982100 Il modulo indirizzato in ***Address*** non è presente.

9982110~1 Il modulo indirizzato non supporta i comandi uscita analogica.

9982150 Il valore da impostare sul modulo analogico non è corretto.

9982200 Il modo di gestione definito in ***Mode*** non è gestito dal modulo.

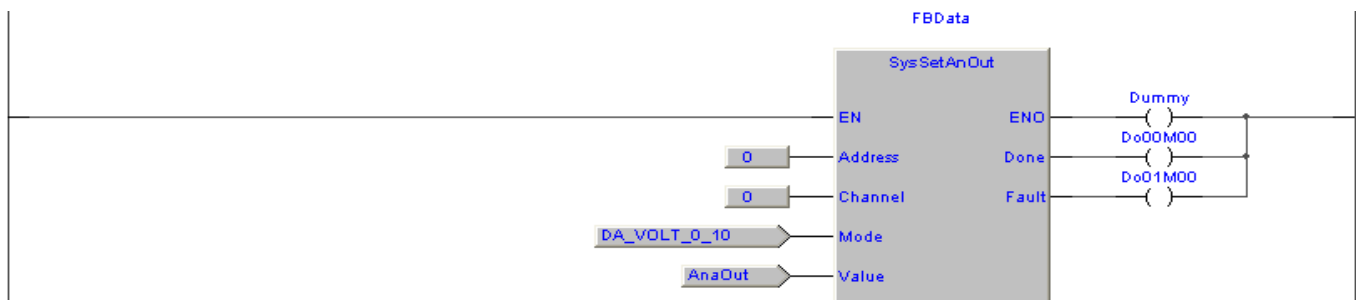
9982210 Errore nella gestione uscita analogica sul modulo.

Esempi

Viene eseguita l'uscita analogica dal canale 0 del modulo 0 in modo 0-10 Volt. Se dato valido viene attivata l'uscita digitale ***Do00M00*** se errore di conversione viene attivata l'uscita digitale ***Do01M00***. Il dato analogico da impostare in uscita nel range da 0.000 a 9.999 è presente nella variabile ***AnaOut***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	SysSetAnOut	Auto	No	0	..	FB SysSetAnAout data
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
3	AnaOut	REAL	Auto	No	0	..	Analog value (Volts)

Esempio LD *(PTP116A100, LD_SysSetAnOut)*

Esempio IL *(PTP116A100, IL_SysSetAnOut)*

```
(* Manage analog output 0 on module 0. *)

    LD   0
    ST   FBData.Address (* Set module address *)

    LD   0
    ST   FBData.Channel (* Set channel *)

    LD   DA_VOLT_0_10
    ST   FBData.Mode (* Set management mode *)

    LD   AnaOut
    ST   FBData.Value (* Store the output value *)

    CAL FBData (* Call the SysSetAnOut function block *)

    LD   FBData.Done
    ST   Do00M00 (* The output is active  if data is set *)

    LD   FBData.Fault
    ST   Do01M00 (* The output is active if execution fault *)
```

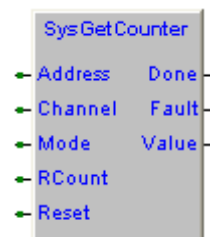
Esempio ST *(PTP116A100, ST_SysSetAnOut)*

```
(* Manage analog output 0 on module 0. *)

    FBData.Value:=AnaOut; (* Store the ouput value *)
    FBData(Address:=0, Channel:=0, Mode:=DA_VOLT_0_10); (* Call the SysSetAnOut function block *)

    Do00M00:=FBData.Done; (* The output is active if data is set *)
    Do01M00:=FBData.Fault; (* The output is active if execution fault *)
```

7.12.6 SysGetCounter, get counter

Type	Library
FB	XTarget_07_0

Questo blocco funzione esegue la lettura di un contatore. Il blocco funzione può essere utilizzato per acquisire il valore del contatore presente sul modulo CPU SlimLine e dai moduli che gestiscono la funzione contatore.

E' prevista la gestione del reset valore di conteggio e della inversione conteggio. In base alla definizione di ***Mode*** è possibile gestire conteggio su fronte positivo, negativo o su entrambi i fronti dell'ingresso clock del contatore. Se il modulo che gestisce il contatore lo prevede è possibile anche definire comandi hardware (Ingressi logici) di reset conteggio e di inversione conteggio.

Address (USINT) Occorre specificare l'indirizzo di modulo da cui eseguire l'acquisizione counter (Range da 0 a 255). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito.
L'indirizzo 255 indica il modulo CPU.

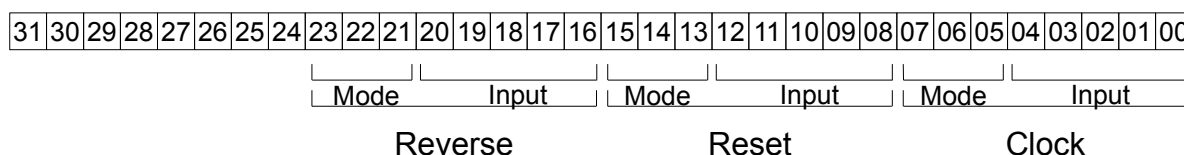
Se viene settato un indirizzo di modulo non presente, si interrompe l'esecuzione e viene settato il bit di ***Fault***.

Channel (USINT) Occorre specificare l'indirizzo del canale sul modulo (Range da 0 a 15).

Se viene settato un indirizzo di canale non presente, si interrompe l'esecuzione e viene settato il bit di ***Fault***.

Mode (UDINT) Modo acquisizione, espresso su 32 bit secondo lo schema riportato.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

Reverse		Reset		Clock	
Mode	Input	Mode	Input	Mode	Input

Clock	Input	Definisce ingresso logico da utilizzare come clock
	Mode	0: Conta su fronte salita 1: Conta su fronte discesa 2: Conta su entrambi i fronti

Reset	Input	Definisce ingresso logico da utilizzare come reset
	Mode	0: Non è utilizzato ingresso di reset 1: Reset counter se ingresso attivo 2: Reset conter se ingresso non attivo

Reverse	Input	Definisce ingresso logico da utilizzare come inversione conteggio
	Mode	0: Non è utilizzato ingresso di reverse 1: Inverte conteggio counter se ingresso attivo 2: Inverte conteggio conter se ingresso non attivo

Per calcolare il valore di mode si applica la formula:

((Reverse mode)*2097152)+((Reverse input)*65536)+((Reset mode)*8192)+((Reset input)*256)+((Clock mode)*32)+(Clock input)

Se viene settato un valore non corretto, si interrompe l'esecuzione e viene settato il bit di ***Fault***.

RCount (BOOL) Reverse counting, attivando questo ingresso ***Value*** viene decrementato ad ogni variazione di conteggio.

Reset (BOOL) Attivando questo ingresso si ha il reset del valore di conteggio ***Value***.

Done (BOOL) Dato counter acquisito, viene attivato per un loop al termine della acquisizione counter.

Fault (BOOL) Errore di acquisizione, viene attivato in caso di errore nella sequenza di acquisizione.

Value (UDINT) Valore contatore.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

9981050 Errore allocazione blocco funzione.

9981060 Terminato spazio memoria rilocabile, non è possibile eseguire l''FB.

9981070 Errore versione blocco funzione.

9981080 Impossibile inizializzare il modulo.

9981100 Il modulo indirizzato in ***Address*** non è presente.

9981110 Il canale definito in ***Channel*** non è gestito.

9981200~4 Il modo di gestione definito in ***Mode*** non è gestito dal modulo.

9981300~2 Errore nella gestione lettura counter dal modulo.

9981990 Non implementata nel simulatore.

Esempi

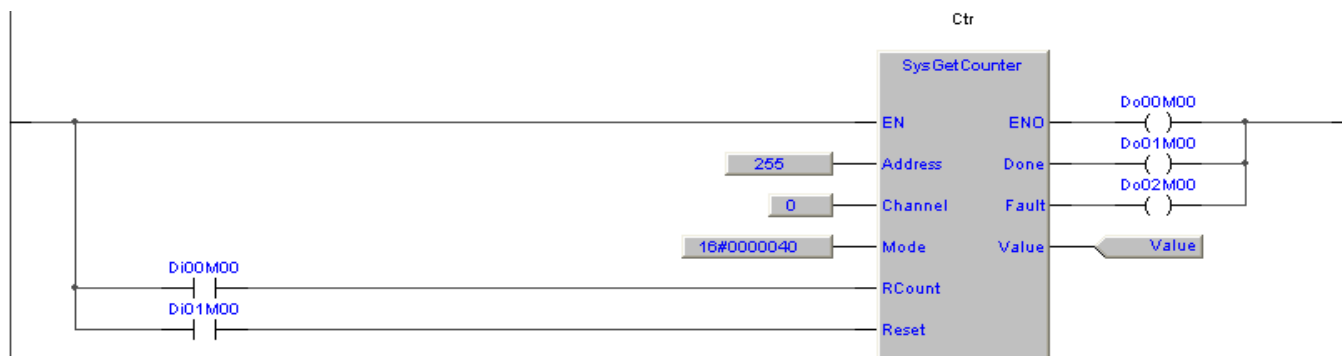
Viene eseguita l'acquisizione del contatore dal modulo CPU di SlimLine, viene eseguito il conteggio su entrambi i fronti dell'ingresso di clock. Il valore di conteggio è trasferito nella variabile ***Value***. Su fine conversione viene attivata l'uscita digitale ***Do01M00*** se errore di conversione viene attivata l'uscita digitale ***Do02M00***.

Attivando l'ingresso ***Di00M00*** viene eseguita l'inversione del conteggio ad ogni variazione dell'ingresso di clock viene decrementato il valore di ***Value*** in uscita.

Attivando l'ingresso ***Di01M00*** viene eseguito il reset del conteggio il valore di ***Value*** in uscita è resettato.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Value	UDINT	Auto	No	0	..	Counter value
2	Ctr	SysGetCounter	Auto	No	0	..	SysGetCounter FB data

Esempio LD

7.12.7 SysGetEncoder, get encoder input

Type	Library
FB	XTarget_07_0

Questo blocco funzione esegue la lettura di un canale encoder. Il blocco funzione può essere utilizzato solo su sistemi che hanno moduli in grado di acquisire encoders incrementali.

E' prevista la gestione della tacca di zero e la possibilità di acquisire valori di quota all'interno di un segnale logico di gate.

Address (USINT) Occorre specificare l'indirizzo di modulo da cui eseguire l'acquisizione encoder (Range da 0 a 15). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito.

Se viene settato un indirizzo di modulo non presente, si interrompe l'esecuzione e viene settato il bit di ***Fault***.

Channel (USINT) Occorre specificare l'indirizzo del canale sul modulo (Range da 0 a 15).

Se viene settato un indirizzo di canale non presente, si interrompe l'esecuzione e viene settato il bit di ***Fault***.

RCount (BOOL) Reverse counting, attivando questo ingresso si inverte l'incremento di quota ***EQuote*** in funzione della direzione di rotazione encoder.

ZeroEn (BOOL) Attivando questo ingresso si ha il reset della quota ***EQuote*** al passaggio della tacca di zero encoder.

GateEn (BOOL) Attivando questo ingresso sul fronte di variazione dell'ingresso di Gate viene trasferito il valore di ***EQuote*** in ***GQuote***.

RGate (BOOL) Attivando questo ingresso viene gestito il fronte di disattivazione dell'ingresso Gate.

Done (BOOL) Dato encoder acquisito, viene attivato per un loop al termine della acquisizione encoder.

Fault (BOOL) Errore di acquisizione, viene attivato in caso di errore nella sequenza di acquisizione.

ZeroD (BOOL) Tacca di zero encoder acquisita, viene settata su acquisizione tacca di zero encoder, si resetta disattivando l'ingresso ***ZeroEn***.

GateD (BOOL) Segnale di Gate acquisito, viene attivato per un loop alla acquisizione del segnale Gate.

GQuote (UINT) Quota di gate, valore di quota encoder ***EQuote*** memorizzata sul fronte selezionato del segnale Gate.

EQuote (UINT) Quota encoder, valore di quota encoder, al raggiungimento del valore minimo o massimo viene eseguito il roll over.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

9980050 Errore allocazione blocco funzione.

9980060 Terminato spazio memoria rilocabile, non è possibile eseguire l'"FB.

9980070 Errore versione blocco funzione.

9980100 Il modulo indirizzato in ***Address*** non è presente.

9980110~2 Il modulo indirizzato non supporta i comandi acquisizione encoder.

9980200 Il modo di gestione definito in ***Mode*** non è gestito dal modulo.

9980210~2 Errore nella gestione lettura encoder dal modulo.

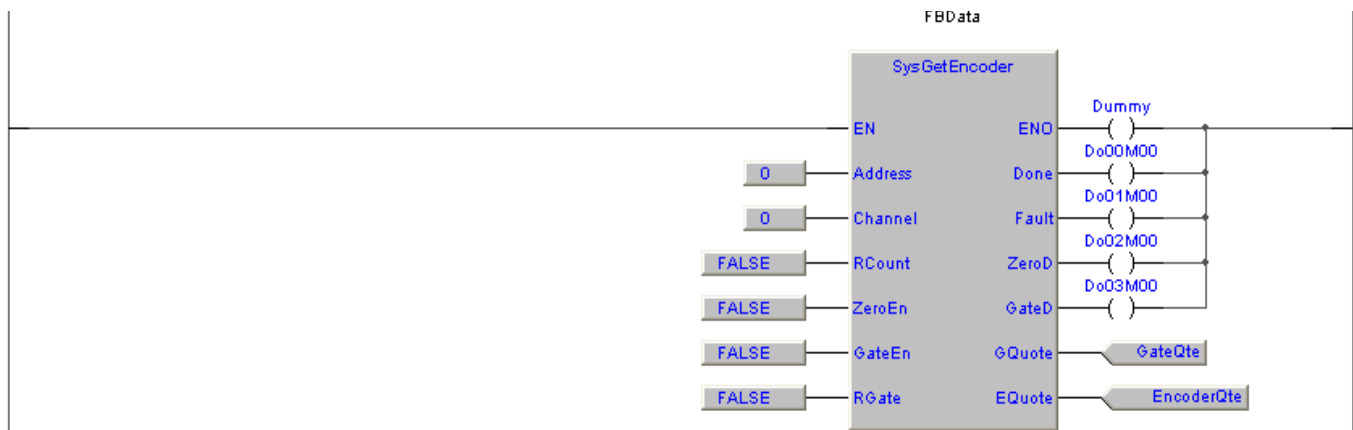
9980990 Non implementata nel simulatore.

Esempi

Viene eseguita l'acquisizione dell'ingresso encoder dal canale 0 del modulo 0, il valore di quota encoder è trasferito nella variabile ***EncoderQte***. Su fine conversione viene attivata l'uscita digitale ***Do00M00*** se errore di conversione viene attivata l'uscita digitale ***Do01M00***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	SysGetEncoder	Auto	No	0	..	FB SysGetEncoder data
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
3	GateQte	UINT	Auto	No	0	..	Gate quote
4	EncoderQte	UINT	Auto	No	0	..	Encoder quote

Esempio LD *(PTP116A100, LD_SysGetEncoder)*

Esempio IL *(PTP116A100, IL_SysGetEncoder)*

```
(* Acquires encoder 0 from module. *)

    LD   0
    ST   FBData.Address (* Set module address *)

    LD   0
    ST   FBData.Channel (* Set channel *)

    LD   FALSE
    ST   FBData.RCount (* Reverse counting *)
    ST   FBData.GateEn (* Gate enable *)
    ST   FBData.RGate (* Reverse gate *)

    CAL  FBData (* Call the SysGetEncoder function block *)

    LD   FBData.Done
    ST   Do00M00 (* The output is active if data is acquired *)

    LD   FBData.Fault
    ST   Do01M00 (* The output is active if execution fault *)

    LD   FBData.GQuote
    ST   GateQte (* Gate quote *)

    LD   FBData.EQuote
    ST   EncoderQte (* Encoder quote *)
```

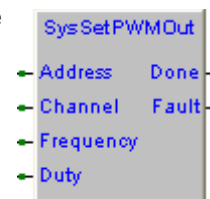
7.12.8 SysSetPWMOut, set PWM output

Type	Library
FB	XTarget_07_0

Questo blocco funzione esegue l'impostazione del valore di frequenza ***Frequency*** e di duty cycle ***Duty*** sul generatore PWM indicato. Il range dei valori di frequenza e duty cycle dipendono dalla scheda utilizzata.

Impostando il valore di ***Frequency*** a 0 viene disabilitato il PWM e l'uscita viene forzata in base al valore di ***Duty***.

Duty < 50: L'uscita viene settata a FALSE.

Duty >= 50: L'uscita viene settata a TRUE.

Address (USINT) Occorre specificare l'indirizzo di modulo da cui eseguire l'acquisizione encoder (Range da 0 a 15). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito. Il valore 255 indica il modulo CPU.
Se viene settato un indirizzo di modulo non presente, si interrompe l'esecuzione e viene settato il bit di ***Fault***.

Channel (USINT) Occorre specificare l'indirizzo del canale sul modulo (Range da 0 a 15).
Se viene settato un indirizzo di canale non presente, si interrompe l'esecuzione e viene settato il bit di ***Fault***.

Frequency (REAL) Valore di frequenza in uscita (Per i limiti fare riferimento al manuale hardware della scheda utilizzata). Il valore è espresso in ***Hz***.

Duty (REAL) Valore di duty cycle del segnale in uscita (Per i limiti fare riferimento al manuale hardware della scheda utilizzata). Il valore è espresso in ***%***.

Done (BOOL) Generatore PWM correttamente impostato.

Fault (BOOL) Errore nella esecuzione.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

9951050 Errore allocazione blocco funzione.

9951070 Errore versione blocco funzione.

9951100 Il modulo indirizzato in ***Address*** non è presente.

9951110 Numero di canale non presente.

9951500 Errore esecuzione.

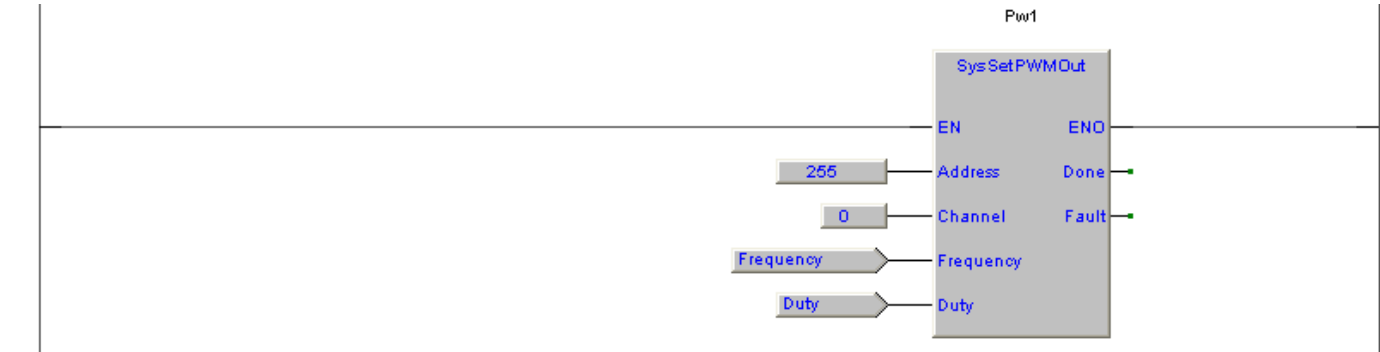
9951990 Non implementata nel simulatore.

Esempi

Viene impostata l'uscita Out 0 del modulo CPU per generare un segnale PWM a 100 Hz con duty cycle 50 %.

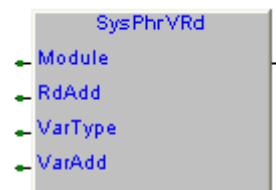
Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Pw1	SysSetPWMOut	Auto	No	0	..	FB data
2	Frequency	REAL	Auto	No	100	..	Frequency set (Hz)
3	Duty	REAL	Auto	No	50	..	DutyCycle set (%)

Esempio LD

7.12.9 SysPhrVRd, read variable from peripheral module

Type	Library
Function	XTarget_07_0

Questa funzione esegue la lettura di una variabile dal modulo periferico di estensione.

Occorre definire l'indirizzo di modulo ***Module***, l'indirizzo della variabile da leggere sul modulo periferico ***RdAdd***, il tipo di variabile ***VarType*** e l'indirizzo del buffer dove trasferire il valore letto ***VarAdd***.

SysPhrVRd
Module
RdAdd
VarType
VarAdd

Parametri funzione:

Module (USINT) Occorre specificare l'indirizzo di modulo da cui eseguire la lettura (Range da 0 a 15). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito.

RdAdd (UINT) Indirizzo variabile da leggere come allocata sul modulo periferico.

VarType (USINT) Tipo variabile, come indicato nella tabella ***Variable types definition***.

VarAdd (UDINT) Indirizzo variabile.

La funzione ritorna:

(BOOL) ***FALSE:*** Errore esecuzione. ***TRUE:*** Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9989100 Il modulo indirizzato in ***Module*** non è presente.

9989110 Il tipo variabile definito in ***VarType*** non è corretto.

9989200 Errore durante l'esecuzione della lettura della variabile dal modulo periferico.

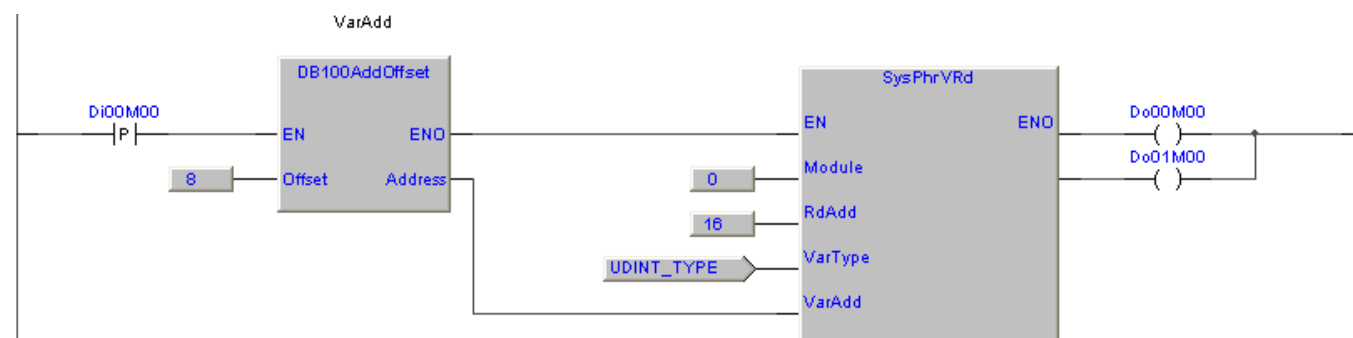
9989990 Non implementata nel simulatore.

Esempi

Attivando l'ingresso ***Di00M00*** viene eseguita la lettura della variabile ***UDINT*** da indirizzo ***16*** dal modulo periferico ***0***. Il valore della variabile è trasferito nella ***DB100*** ad offset ***8***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarAdd	DB100AddOffset	Auto	No	0	..	Variable address calculation

Esempio LD

7.12.10 SysPhrVWr, write variable to peripheral module

Type	Library
Function	XTarget_07_0

Questa funzione esegue la scrittura di una variabile sul modulo periferico di estensione.

Occorre definire l'indirizzo di modulo ***Module***, l'indirizzo della variabile da scrivere sul modulo periferico ***WrAdd***, il tipo di variabile ***VarType*** e l'indirizzo del buffer dove si trova il valore da scrivere ***VarAdd***.

SysPhrVWr
Module
WrAdd
VarType
VarAdd

Parametri funzione:

Module (USINT) Occorre specificare l'indirizzo di modulo su cui eseguire la scrittura (Range da 0 a 15). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito.

WrAdd (UINT) Indirizzo variabile da scrivere come allocata sul modulo periferico.

VarType (USINT) Tipo variabile, come indicato nella tabella ***Variable types definition***.

VarAdd (UDINT) Indirizzo variabile.

La funzione ritorna:

(BOOL) ***FALSE:*** Errore esecuzione. ***TRUE:*** Funzione eseguita correttamente.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9988100 Il modulo indirizzato in ***Module*** non è presente.

9988110 Il tipo variabile definito in ***VarType*** non è corretto.

9988200 Errore durante l'esecuzione della scrittura della variabile sul modulo periferico.

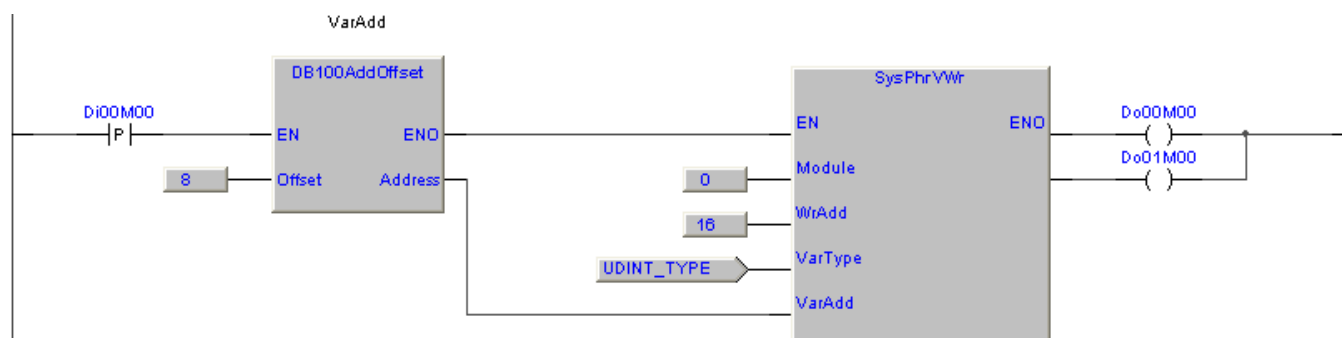
9988990 Non implementata nel simulatore.

Esempi

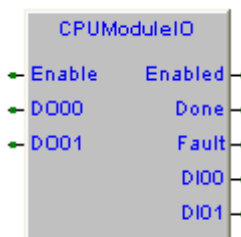
Attivando l'ingresso ***Di00M00*** viene eseguita la scrittura della variabile ***UDINT*** ad indirizzo ***16*** sul modulo periferico ***0***. Il valore da scrivere è presente nella ***DB100*** ad offset ***8***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarAdd	DB100AddOffset	Auto	No	0	..	Variable address calculation

Esempio LD

7.12.11 CPUModuleIO, CPU module I/O management

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione esegue la lettura dei due ingressi digitali e la gestione delle due uscite digitali presenti sul modulo CPU dello SlimLine.

Enable (BOOL) Abilitazione gestione I/O, attivando l'ingresso vengono acquisiti gli ingressi logici e gestite le uscite logiche. Disattivando l'ingresso vengono disattivate anche le uscite logiche sul modulo CPU.

DO00 (BOOL) Stato della uscita logica DO00 presente sul modulo CPU.

DO01 (BOOL) Stato della uscita logica DO01 presente sul modulo CPU.

Enabled (BOOL) Blocco funzione abilitato.

Done (BOOL) Acquisizione input e gestione output eseguita.

Fault (BOOL) Errore su acquisizione input e gestione output.

DI00 (BOOL) Stato dell'ingresso logico DI00 presente sul modulo CPU.

DI01 (BOOL) Stato dell'ingresso logico DI01 presente sul modulo CPU.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10001100 Errore su esecuzione funzione lettura ingressi logici da modulo CPU

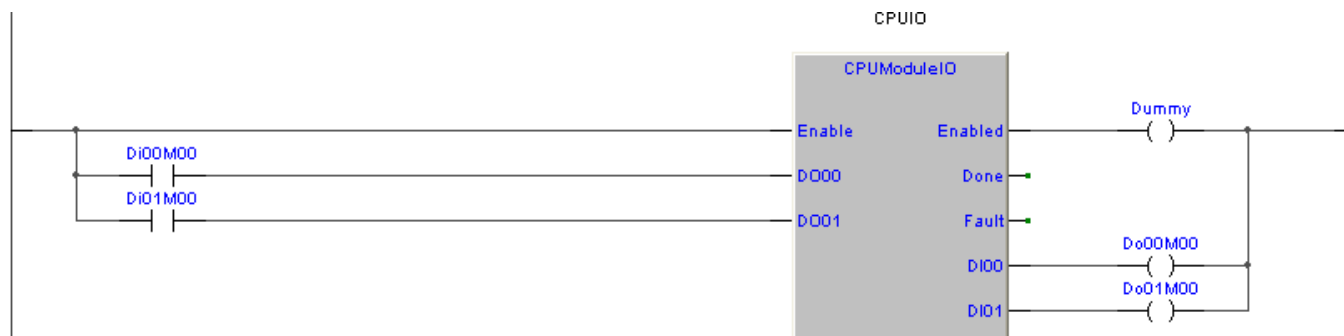
10001200 Errore su esecuzione funzione di set uscite logiche su modulo CPU

Esempi

Viene eseguita la lettura dei due ingressi digitali e la gestione delle due uscite digitali presenti sul modulo CPU. Lo stato dei due ingressi logici è copiato nelle uscite ***Do00M00*** e ***Do01M00***. Lo delle variabili ***Di00M00*** e ***Di01M00*** è trasferito sulle due uscite logiche.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	CPUIO	CPUModuleIO	Auto	No	0	..	CPUModuleIO function block
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable

Esempio LD *(PTP114A200, LD_CPUModuleIO)*

7.12.12 SysI2CWrRd, writes/reads on I2C extension bus

Type	Library
Function	XTarget_07_0

SysI2CWrRd
Address
WrBytes
WrBuffer
RdBytes
RdBuffer

Questa funzione gestisce la scrittura/lettura sul bus I2C di estensione. L'utilizzo di questa funzione permette di poter gestire qualsiasi componente I2C connesso al bus di estensione.

Attenzione, siccome il bus I2C è utilizzato per l'accesso ai moduli di estensione per non rallentarne l'accesso si consiglia di eseguire comandi I2C non più lunghi di 4 bytes in scrittura e lettura. Nel caso serva gestire più bytes si consiglia di spezzare il comando in più comandi consecutivi.

Parametri funzione:

Address (USINT) Indirizzo dispositivo I2C range da 16#00 a 16#1F, da 16#30 a 16#7F.

WrBytes (USINT) Numero di bytes dati da scrivere. 0 se solo lettura.

WrBuffer (@USINT) Indirizzo buffer memoria che contiene i dati da scrivere. NULL se solo lettura.

RdBytes (USINT) Numero di bytes dati da leggere. 0 se solo scrittura.

RdBuffer (@USINT) Indirizzo buffer memorizzazione dati letti. NULL se solo scrittura.

La funzione ritorna:

(BOOL) ***TRUE:*** Comando eseguito correttamente.

Codici di errore

In caso di errore la funzione torna ***FALSE*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

9953100 Indirizzo I2C fuori range (Valore superiore a 16#7F).

9953105 Indirizzo I2C utilizzato da un modulo di estensione.

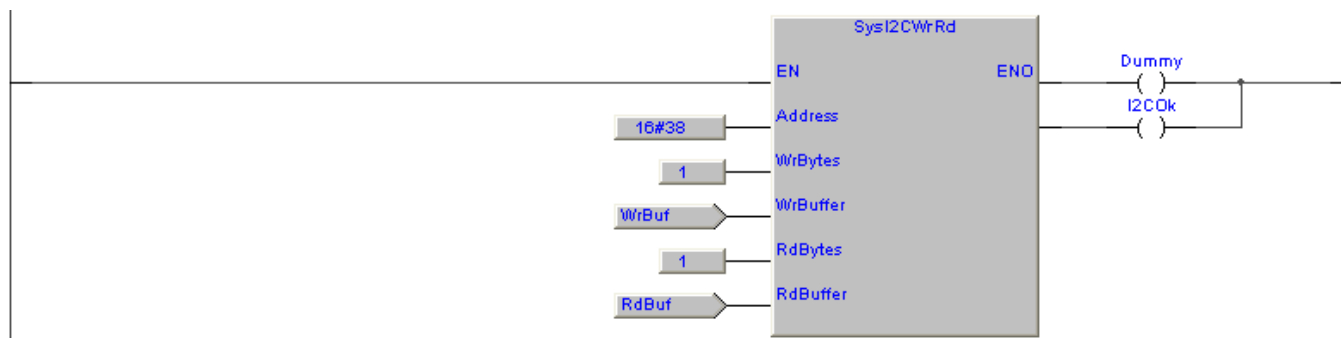
9953990 Non implementata nel simulatore.

Esempi

E' riportato un semplice programma che esegue la lettura e scrittura di un chip I/O expander PCF9670 su bus I2C. Il chip ha indirizzo 16#38.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
2	I2COk	BOOL	Auto	No	FALSE	..	I2C Ok
3	RdBuf	BYTE	Auto	No	0	..	Read data buffer
4	WrBuf	BYTE	Auto	No	0	..	Write data buffer

Esempio LD

7.12.13 StrainGaugeAcq, strain gauge acquisition

Type	Library
FB	ePLCUtyLib_C000

StrainGaugeAcq
Enable — Enabled
DoOfsCalibration — Done
DoFSCalibration — Fault
Address — Value
Channel
Mode
FilterCf
FSReference
SGaugeFullScale
SGaugeRatio
OfsCalibration
FSCalibration

Questo blocco funzione gestisce l'acquisizione da dispositivi strain gauge, all'interno viene gestito l'FB di acquisizione analogica ***SysGetAnInp*** per acquisire l'ingresso analogico di lettura strain gauge. In ***Address*** e ***Channel*** vanno definiti indirizzo ed il canale del modulo analogico di acquisizione. In ***Mode*** occorre definire il modo di acquisizione approppriato allo strain gauge da acquisire.

E' importante effettuare la taratura della acquisizione (I valori di taratura sono memorizzati nelle variabili ***OfsCalibration*** e ***FSCalibration***). Per la taratura dell'offset collegare lo strain gauge e senza nessun peso applicato attivare per un loop ***DoOfsCalibration***. Per la taratura del fondo scala applicare allo strain gauge un peso il più possibile prossimo al valore di fondo scala. Il valore del peso applicato va definito in **FSReference**, dopo un tempo di stabilizzazione lettura peso attivare per un loop ***DoFSCalibration***.

In **Mode** occorre scegliere un guadagno il più alto possibile, ma facendo in modo che: (**SGaugeRatio x Alim. x Mode)** < **Alim.** dove **Alim.** è la tensione di alimentazione dello strain gauge. Esempio: strain gauge alimentato a 5V e con **SGaugeRatio** = 2mV/V. Scegliendo **Mode** = AD_VIN_VREF_G_128 si avrà (2mV/V * 5V * 128) = 1280mV quindi ben sotto i 5V e quindi ok.

In ***FilterCf*** è possibile definire un coefficente di filtro sulla acquisizione del peso, più il valore è grande e più il valore di peso è filtrato. Con 0, non c'è filtro lasciando la possibilità di inserire un proprio filtro personalizzato esterno.

Enable (BOOL)	Abilitazione blocco funzione.
DoOfsCalibration (BOOL)	Comando calibrazione offset.
DoFSCalibration (BOOL)	Comando calibrazione fondo scala.
Address (USINT)	Occorre specificare l'indirizzo di modulo da cui eseguire l'acquisizione dello strain gauge (Range da 0 a 15). Il valore 0 indica il primo modulo di estensione, 1 il secondo e così di seguito.
Channel (USINT)	Occorre specificare l'indirizzo del canale sul modulo (Range da 0x00 a 0x0F). Se viene settato un indirizzo di canale non presente, si interrompe l'esecuzione e viene settato il bit di ***Fault***.
Mode (USINT)	Occorre specificare il modo di acquisizione analogica, ***Analog to digital mode***.
FilterCf (REAL)	Valore del coefficiente di filtro acquisizione (Nr)
FSReference (REAL)	Valore riferimento utilizzato per la taratura del fondo scala dello strain gauge (Nr)
SGaugeFullScale (REAL)	Valore di fondo scala dello strain gauge (Caratteristica dello strain gauge) (Nr)
SGaugeRatio (REAL)	Valore del rapporto mv/V dello strain gauge (Caratteristica dello strain gauge) (Nr)
OfsCalibration (@REAL)	Indirizzo variabile di appoggio valore di calibrazione offset (Deve essere RETAIN)
FSCalibration (@REAL)	Indirizzo variabile di appoggio valore di calibrazione fondo scala (Deve essere RETAIN)
Enabled (BOOL)	Blocco funzione abilitato.
Done (BOOL)	Dato acquisito, viene attivato al termine della acquisizione strain gauge.
Fault (BOOL)	Errore di acquisizione, viene attivato in caso di errore nella sequenza di acquisizione.
Value (REAL)	Ritorna il valore di acquisizione espresso nella unità utilizzata per definire ***FSReference*** e ***SGaugeFullScale***.

Codici di errore

In caso di errore viene attivato per un loop ***Fault*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

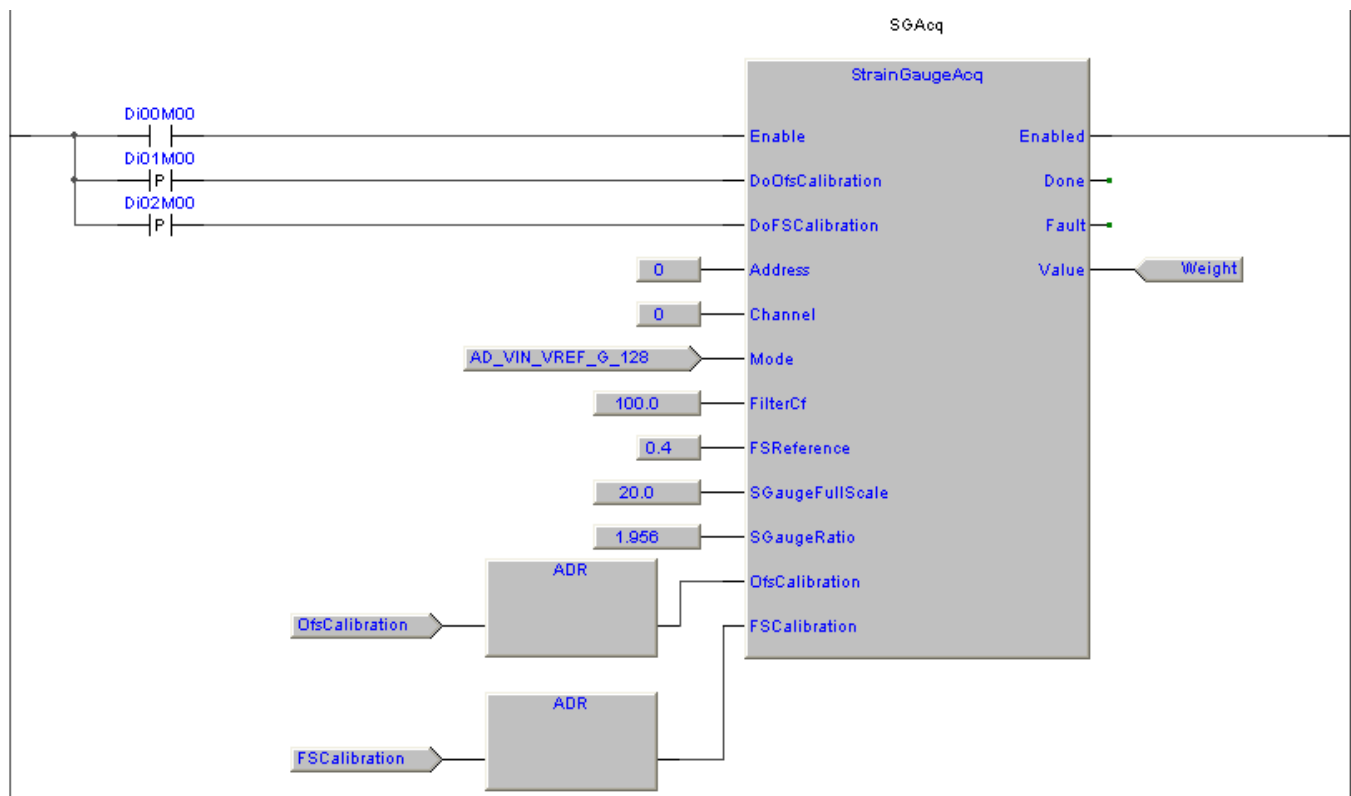
10045100 Valore di ***Mode*** acquisizione non corretto.

Esempi

Nell'esempio viene acquisita una cella di carico con fondo scala da 20 Kg e 1.956 mv/V. La cella viene tarata con un peso di riferimento da 400 grammi. Attivando ***Di00M00*** viene abilitata l'acquisizione, gli ingressi ***Di01M00*** e ***Di02M00*** permettono di calibrare la cella (Attenzione le variabili ***OfsCalibration*** e ***FSCalibration*** devono essere definite RETAIN). Nella variabile ***Weight*** viene ritornato il peso. Avendo definito ***SGaugeFullScale*** in Kg anche ***Value*** sarà espresso in Kg.

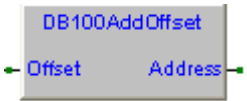
Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Weight	REAL	Auto	No	0	..	Weight (Kg)
2	SGAcq	StrainGaugeAcq	Auto	No	0	..	Strain gauge acquisition

Esempio LD

7.13 Funzioni ed FB di utilità generale

7.13.1 DB100AddOffset, returns DB100 address offset

Type	Library
FB	ePLCUtyLib_C000

Questa funzione ritorna l'indirizzo della locazione di memoria all'interno della ***DB100***, il cui offset è fornito come parametro.

Parametri funzione:

Offset (UINT) Offset locazione memoria

La funzione ritorna:

(UDINT) Indirizzo della locazione di memoria

Esempi

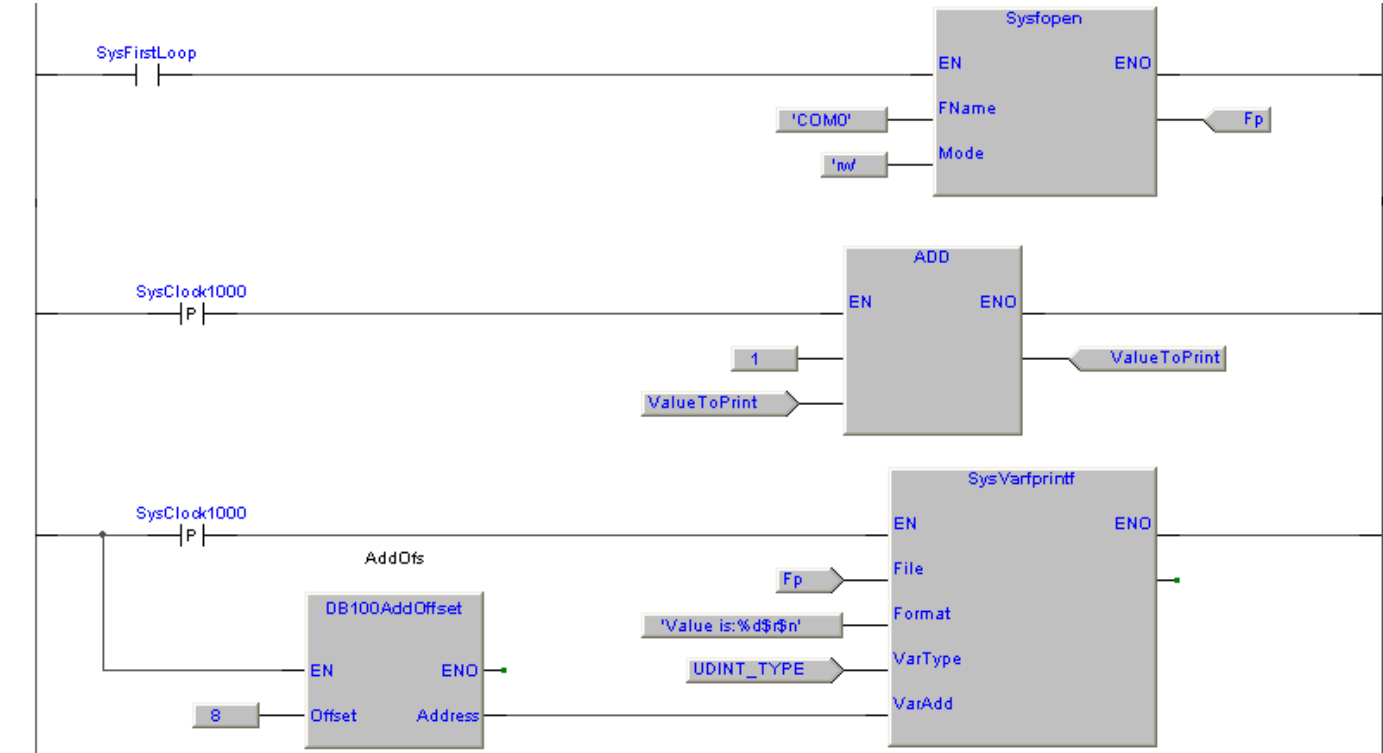
La variabile ***ValueToPrint*** è allocata nella ***DB100*** ad offset ***8***, ogni secondo ne viene incrementato il valore e su porta seriale ***COM0*** viene inviata la stringa ***Value is: xxxx***.

Definizione variabili globali

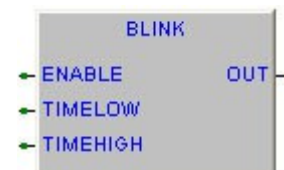
	Name	Type	Address	Group	Array	Init value	Attribute	Description
1	ValueToPrint	UDINT	%MD100.8		No	0	..	Value to print

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	AddOfs	DB100AddOffset	Auto	No	0	..	Address offset calculation

Esempio LD

7.13.2 BLINK, blink command

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione gestisce una uscita lampeggiante con tempo di ciclo definibile. Attivando l'ingresso ***ENABLE*** l'uscita ***OUT*** inizia a lampeggiare con tempi di ciclo alto e basso definiti.

ENABLE (BOOL) Abilitazione blocco funzione, attivandolo viene gestita l'uscita OUT lampeggiante. Disattivandolo l'uscita OUT viene resettata.

TIMELOW (UDINT) Definisce il tempo in cui l'uscita OUT rimane nello stato logico low, espresso in mS.

TIMEHIGH (UDINT) Definisce il tempo in cui l'uscita OUT rimane nello stato logico high espresso in mS.

OUT (BOOL) Stato uscita lampeggiante.

Esempi

Viene impostato un lampeggio con 100 mS di tempo On e 1000 mS di tempo Off. Attivando l'ingresso digitale ***Di00M00*** l'uscita digitale ***Do00M00*** lampeggia con i tempi definiti.

Disattivando l'ingresso digitale ***Di00M00*** l'uscita digitale ***Do00M00*** si azzera immediatamente.

Definizione variabili

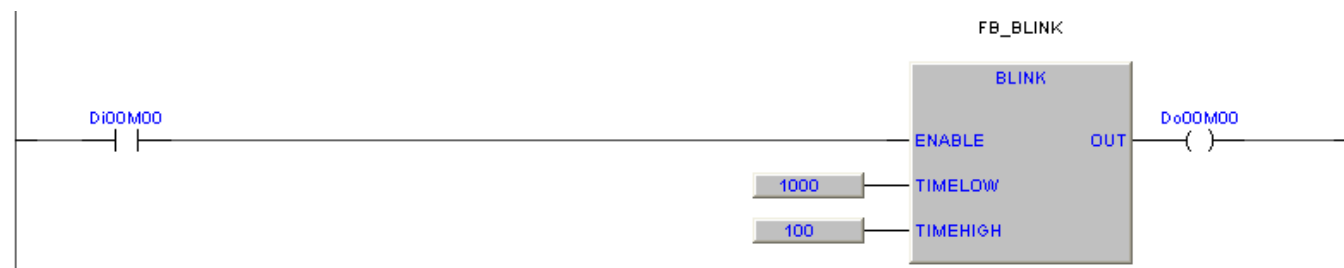
	Name	Type	Address	Array	Init value	Attribute	Description
1	FBeBLINK	BLINK	Auto	No	0	..	eBLINK (Blink out function block)

Esempio LD *(PTP14A100, LD_BLINK)*

FB_BLINK

BLINK

Di00M00 — ENABLE — OUT — Do00M00

1000 — TIMELOW

100 — TIMEHIGH

Esempio IL *(PTP14A100, IL_BLINK)*

```
CAL FB_BLINK (* Call the BLINK function block *)

 LD  Di00M00
 ST  FB_BLINK.ENABLE (* Transfer the digital input to enable input *)

 LD  1000
 ST  FB_BLINK.TIMELOW (* Set the time low *)

 LD  100
 ST  FB_BLINK.TIMEHIGH (* Set the time high *)

 LD  FB_BLINK.OUT
 ST  Do00M00 (* Copy FB output to logic output *)
```

Esempio ST *(PTP14A100, ST_BLINK)*

```
FB_BLINK(TIMELOW:=1000, TIMEHIGH:=100); (* Call the BLINK function block *)

FB_BLINK.ENABLE:=Di00M00; (* Transfer the digital input to FB enable *)
Do00M00:=FB_BLINK.OUT; (* Transfer the FB output to digital output *)
```

7.13.3 BlinkValue, blink out value

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione gestisce una uscita lampeggiante con possibilità di definire il numero di lampeggi. Attivando l'ingresso ***Enable*** e definendo il numero di lampeggi in ***BlinkNr***, l'uscita ***OutBit*** inizia a lampeggiare con il numero di lampeggi definito.

Il numero di lampeggi è definito in decine ed unità, il valore delle decine è riportato con un lampeggio lento (1 Sec), mentre il numero delle unità è riportato con un lampeggio veloce (250 mS). Una pausa di 3 Sec separa le sequenze di lampeggio.

Enable (BOOL) Abilitazione blocco funzione, attivandolo viene gestita l'uscita ***OutBit*** lampeggiante. Disattivandolo l'uscita viene resettata.

BlinkNr (USINT) Definisce il numero di lampeggi dell'uscita ***OutBit***. Definendo tempo 0 l'uscita si disattiva.

OutBit (BOOL) Stato uscita lampeggiante.

Esempi

Attivando l'ingresso digitale ***Di00M00*** l'uscita digitale ***Do00M00*** lampeggia con 2 lampeggi lenti (1 Sec), 3 lampeggi veloci (250 mS) ed una pausa di 3 Sec.

Disattivando l'ingresso digitale ***Di00M00*** l'uscita digitale ***Do00M00*** si azzera immediatamente.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	BlinkValue	Auto	No	0	..	FB BlinkValue data

Esempio LD *(PTP14A100, LD_BlinkValue)*

Esempio IL *(PTP14A100, IL_BlinkValue)*

```
CAL FBData (* Call the "BlinkValue" function block *)

LD  Di00M00
ST  FBData.Enable (* Transfer the digital input to enable input *)

LD  23
ST  FBData.BlinkNr (* Set the number of blink *)

LD  FBData.OutBit
ST  Do00M00 (* Copy FB output to logic output *)
```

Esempio ST *(PTP14A100, ST_BlinkValue)*

```
FBData(BlinkNr:=23); (* Call the BLINK function block *)

FBData.Enable:=Di00M00; (* Transfer the digital input to FB enable *)
Do00M00:=FBData.OutBit; (* Transfer the FB output to digital output *)
```

7.13.4 ModbusMaster, modbus master

Type	Library
FB	ePLCUtyLib_C000

ModbusMaster
Enable Done
SpyOn Ok
File Fault
Type Errors
Node
FCode
Address
Points
Buffer
IFTime
Timeout
Delay

Questo blocco funzione esegue la gestione del protocollo modbus master, con ***Type*** è possibile selezionare il tipo di protocollo RTU, Ascii ed over IP. Con ***File*** è possibile definire il terminale di I/O su cui effettuare la comunicazione.

Attivando ***Enable*** sul terminale di I/O viene inviato un frame per eseguire la funzione modbus definita in ***Fcode*** sul nodo definito in Node. Terminata l'esecuzione del comando viene attivata l'uscita ***Done***. Se l'esecuzione comando ha esito positivo si attiva per un loop di programma l'uscita ***Ok***. Disattivando ***Enable*** si azzera ***Done*** e l'eventuale ***Fault***, per eseguire nuovamente il comando occorre riabilitare l'ingresso ***Enable***. L'ingresso ***SpyOn*** se attivo permette di spiare il funzionamento della FB.

Se ***FCode*** è una funzione di lettura, il valore delle variabili a partire dall'indirizzo definito in ***Address*** per il numero di variabili definito da ***Points***, viene letto dal sistema slave e trasferito nelle variabili indirizzate da ***Buffer***.

Se ***FCode*** è una funzione di scrittura, il valore delle variabili presenti nel buffer di memoria indirizzato da ***Buffer*** per il numero di variabili definito da ***Points***, è inviato al dispositivo slave che lo trasferirà nelle sue variabili a partire dall'indirizzo definito in ***Address***.

In caso di errore esecuzione o tempo di esecuzione comando superiore al tempo definito in ***Timeout***, viene attivata per un loop di programma l'uscita ***Fault*** ed incrementato il valore in ***Errors***.

Enable (BOOL) Comando abilitazione esecuzione comando modbus. Per rieseguire il comando disabilitare e poi riabilitare questo ingresso.

SpyOn (BOOL) Se attivo permette di spiare il funzionamento della FB.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

Type (USINT) Tipo di protocollo modbus. 0:RTU, 1:Ascii, 2:TCP

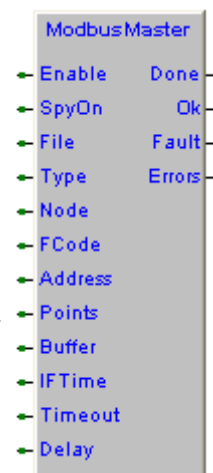
Node (USINT) Numero di nodo modbus su cui effettuare il comando (Range da 0 a 255).

FCode (USINT) Codice funzione modbus da eseguire nel comando (Range da 0 a 255).

Codice	Descrizione
16#01	Read coil status (Massimo 250 coils)
16#02	Read input status (Massimo 125 inputs)
16#03	Read holding registers (Massimo 125 registri)
16#04	Read input registers (Massimo 125 registri)
16#06	Preset single register
16#0F	Force multiple coils (Massimo 250 coils)
16#10	Preset multiple registers (Massimo 32 registri)
16#41	Read memory bytes (User function) (Massimo 250 bytes)
16#42	Write memory bytes (User function) (Massimo 250 bytes)

Address (UINT) Indirizzo di allocazione variabili su sistema slave. In accordo alle specifiche modbus l'indirizzo inviato nel frame dati è (***Address-1***) (Range da 16#0001 a 16#FFFF).

Points (USINT) Numero di variabili consecutive su cui opera il comando.

Buffer (@USINT) Indirizzo buffer dati letti o da scrivere.

IFTime (UDINT) Tempo ricezione caratteri (µS), se comunicazione su porta seriale il tempo deve essere definito in base al baud rate. Nel caso di comunicazione su rete ethernet è possibile definire il valore minimo.

Baud rate	Tempo
300	112000
600	56000
1200	28000
2400	14000
4800	7000
9600	3430

Baud rate	Tempo
19200	1720
38400	860
57600	573
76800	429
115200	286

Timeout (UINT) Tempo massimo esecuzione comando espresso in mS. Se il comando non termina nel tempo definito viene abortito ed attivata l'uscita ***Fault***.

Delay (UINT) Tempo di pausa dopo l'esecuzione del comando modbus espresso in mS.

Done (BOOL) Si attiva al termine della esecuzione comando.

Ok (BOOL) Attivo per un loop se esecuzione comando corretta.

Fault (BOOL) Attivo per un loop se errore esecuzione comando.

SpyTrg (UDINT) Trigger per FB SpyDataFile.

SpyPtr (@USINT) Puntatore a buffer dati da spiare.

Errors (UDINT) Numero di errori, incrementato ad ogni nuovo errore, raggiunto valore massimo riparte da 0.

Trigger di spy

Se ***SpyOn*** attivo viene eseguita la funzione ***SysSpyData*** che permette di spiare il funzionamento della FB. Sono previsti vari livelli di triggers.

TFlags	Descrizione
16#00000001	***Tx:*** Invio frame comando modbus.
16#00000002	***Rx:*** Ricezione frame risposta modbus.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

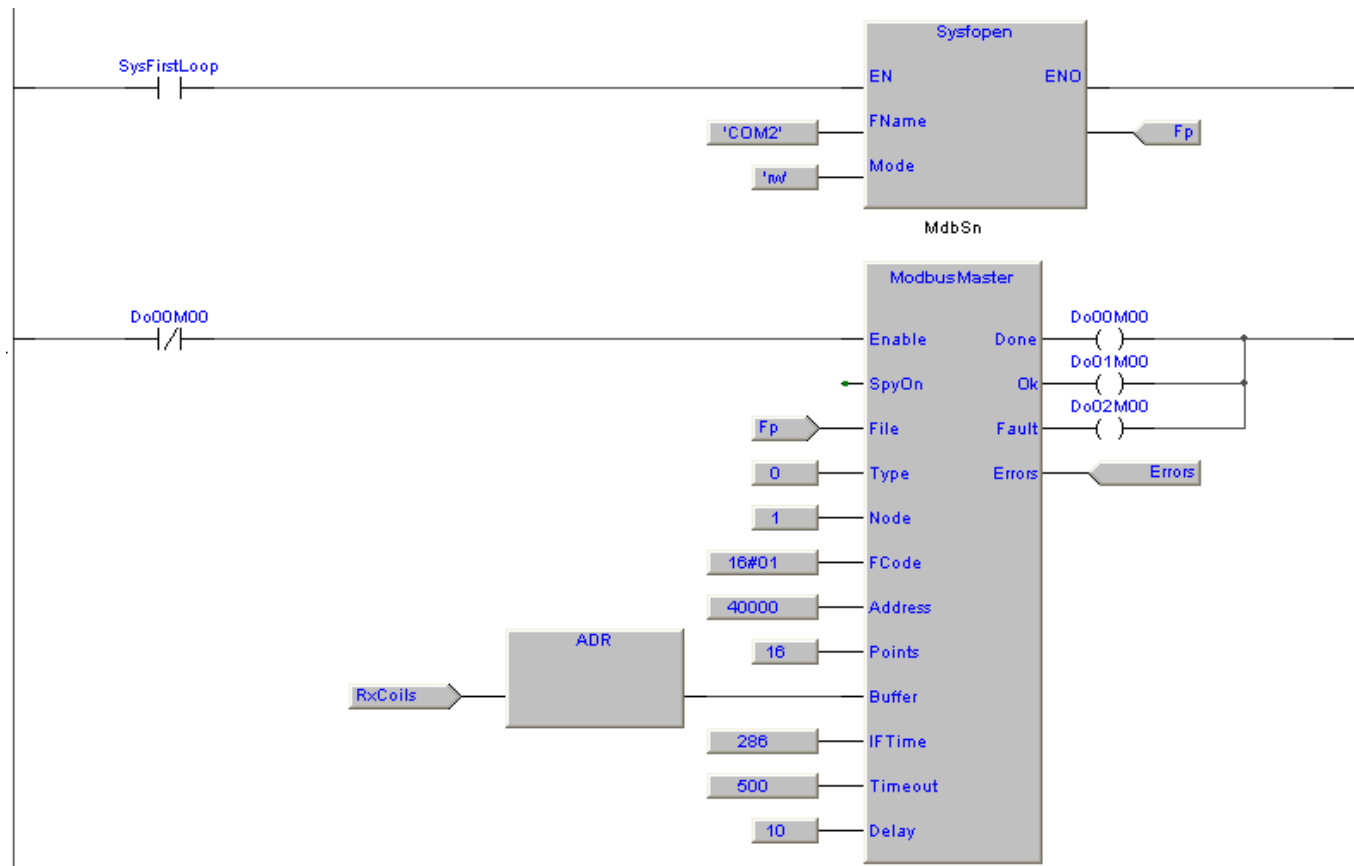
10007010	Valore di ***File*** non definito.
10007050	Timeout esecuzione.
10007060	Errore esecuzione.
10007080	Errore definizione ***Type***.
10007100	Codice funzione definito in ***Function*** non gestito.
10007120	Valore di ***Points*** errato.
10007200~2	Errore trasmissione frame comando.
10007500~7	Errore in ricezione frame risposta (Carattere errato, lunghezza errata, CRC).

Esempi

Viene presentato un esempio di lettura da un sistema SlimLine slave. Viene eseguita la lettura di 16 coils a partire da indirizzo 16#01 dal nodo modbus 1. Il valore dei coils letti è trasferito nell'array ***RxCoils***. Terminata la lettura si attiverà per un loop l'uscita logica ***Do01M00***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RxCoils	BOOL	Auto	[0..15]	16(0)	..	Rx coils data buffer
2	Errors	UDINT	Auto	No	0	..	Modbus communication errors
3	Fp	FILEP	Auto	No	0	..	File pointer
4	MdbSn	ModbusMaster	Auto	No	0	..	Modbus master FB

Esempio LD *(PTP114A620, LD_ModbusMaster)*

7.13.5 ModbusSlave, modbus slave

Type	Library
FB	ePLCUtyLib_C000

Sui sistemi SlimLine il protocollo modbus slave è già implementato dal sistema operativo, pertanto non occorre inserire blocchi funzione appositi nel programma utente. Questo blocco esegue l'override della gestione di sistema operativo e si utilizza in casi particolari, dove non è possibile utilizzare la gestione implementata nel sistema operativo. Per esempio quando si vuole consentire l'accesso ad un propria area di memoria diversa dalla DB100.

Questo blocco funzione esegue la gestione del protocollo modbus slave con ***Type*** è possibile selezionare il tipo di protocollo RTU, Ascii ed over IP. Con ***File*** è possibile definire il terminale di I/O su cui effettuare la comunicazione.

Occorre definire il nodo modbus ***Node***, e l'eventuale offset di indirizzo frame modbus ***Offset***. I comandi modbus ricevuti operano sul buffer di memoria il cui indirizzo è definito in ***Buffer*** e la dimensione in bytes è definita in ***Size***.

In ***IFTime*** occorre definire il tempo di interframe dei comandi modbus, cioè il tempo che intercorre tra la ricezione di un comando ed il comando successivo. Su linea seriale questo tempo coincide con il tempo di ricezione di 3 caratteri al baud rate definito.

Alla ricezione di ogni comando modbus corretto si attiva per un loop l'uscita ***Done***, in caso di errore comando viene attivata per un loop l'uscita ***Fault*** ed incrementato il valore in ***Errors***.

Enable (BOOL) Comando di abilitazione blocco funzione.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

Type (USINT) Tipo di protocollo modbus. 0:RTU, 1:Ascii, 2:TCP

Node (USINT) Numero di nodo modbus (Range da 0 a 255).

Offset (UINT) Offset su indirizzo modbus ricevuto nel frame dati (Range da 16#0000 a 16#FFFF).

Buffer (@USINT) Indirizzo buffer dati su cui operano i comandi modbus.

Size (UINT) Dimensione in byte del buffer dati su cui operano i comandi modbus.

IFTime (UDINT) Tempo ricezione caratteri (µS), se comunicazione su porta seriale il tempo deve essere definito in base al baud rate. Nel caso di comunicazione su rete ethernet è possibile definire il valore minimo.

Baud rate	Tempo
300	112000
600	56000
1200	28000
2400	14000
4800	7000
9600	3430

Baud rate	Tempo
19200	1720
38400	860
57600	573
76800	429
115200	286

Done (BOOL) Attivo per un loop alla ricezione di comando modbus.

Fault (BOOL) Attivo per un loop su errore ricezione comando modbus.

Errors (UDINT) Numero di errori riscontrati. Viene incrementato ad ogni nuovo errore, raggiunto il valore massimo il conteggio riparte da 0.

Comandi supportati

Il blocco funzione supporta solo alcuni comandi previsti dal protocollo modbus, i comandi supportati sono:

Codice	Descrizione
16#01	Read coil status (Massimo 250 coils)
16#02	Read input status (Massimo 250 coils)
16#03	Read holding registers (Massimo 125 registri)
16#04	Read input registers (Massimo 125 registri)
16#05	Force single coil
16#06	Preset single register
16#08	Loopback diagnostic test
16#0F	Force multiple coils (Massimo 250 coils)
16#10	Preset multiple registers (Massimo 125 registri)
16#41	Read memory bytes (User function) (Massimo 250 bytes)
16#42	Write memory bytes (User function) (Massimo 250 bytes)

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore. In caso di eccezione su comando modbus viene riportato il codice di errore ma non viene attivata l'uscita ***Fault***.

10038010 Valore di ***File*** non definito.

10038050 Errore di timeout esecuzione.

10038060 Errore esecuzione.

10038080 Errore definizione ***Type***.

10038100~10 Errore in ricezione frame (Carattere errato, lunghezza errata).

10038200~2 Errore trasmissione frame risposta.

10038501 Eccezione 01.***Illegal function***, comando ricevuto non è tra quelli gestiti.

10038502 Eccezione 02. ***Illegal data address***, comando ricevuto ha indirizzo o numero dati fuori range.

10038503 Eccezione 03. ***Illegal data value***, comando ricevuto ha campo dati fuori range.

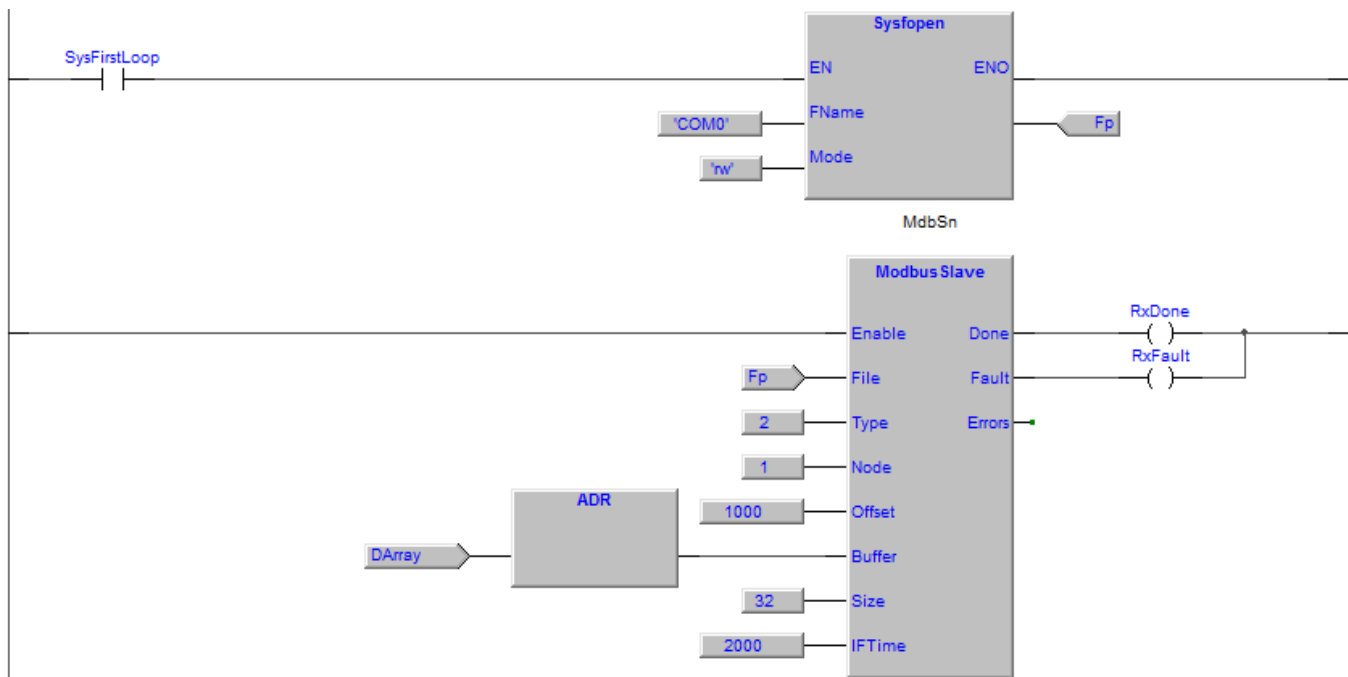
10038504 Eccezione 04.***Failure in associated device***, comando ricevuto contiene imprecisioni.

Esempi

Viene gestito il protocollo modbus slave su porta seriale ***COM0***, si utilizza le impostazioni seriali di default ***115200, e, 8, 1***. I comandi modbus possono agire sull'array di WORD ***DArray***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RxDone	BOOL	Auto	No	FALSE	..	Modbus Rx command Ok
2	RxFault	BOOL	Auto	No	FALSE	..	Modbus Rx command fault
3	Fp	FILEP	Auto	No	0	..	File pointer
4	MdbSn	ModbusSlave	Auto	No	0	..	Modbus RTU slave FB
5	DArray	WORD	Auto	[0..31]	32(0)	..	Modbus accessible area

Esempio LD *(PTP114A610, LD_ModbusSlave)*

7.13.6 ONOFFCYCLE, on/off cycle with random times

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione esegue la temporizzazione di un ciclo On/Off con tempi random di On e di Off definibili tra valori minimo e massimo.

Attivando il comando di ***Enable*** l'uscita ***Out*** esegue un lampeggio On/Off con tempi random compresi tra i valori minimo e massimo definiti. Disabilitando l'ingresso l'uscita Out si disattiva.

La variabile ***Delay*** ritorna il valore di ritardo attualmente attivo.

Enable (BOOL) Comando di abilitazione.

MinOffTime (UDINT) Valore minimo di tempo off comando (mS).

MaxOffTime (UDINT) Valore massimo di tempo off comando (mS).

MinOnTime (UDINT) Valore minimo di tempo on comando (mS).

MaxOnTime (UDINT) Valore massimo di tempo on comando (mS).

Out (BOOL) Stato comando On/Off in uscita.

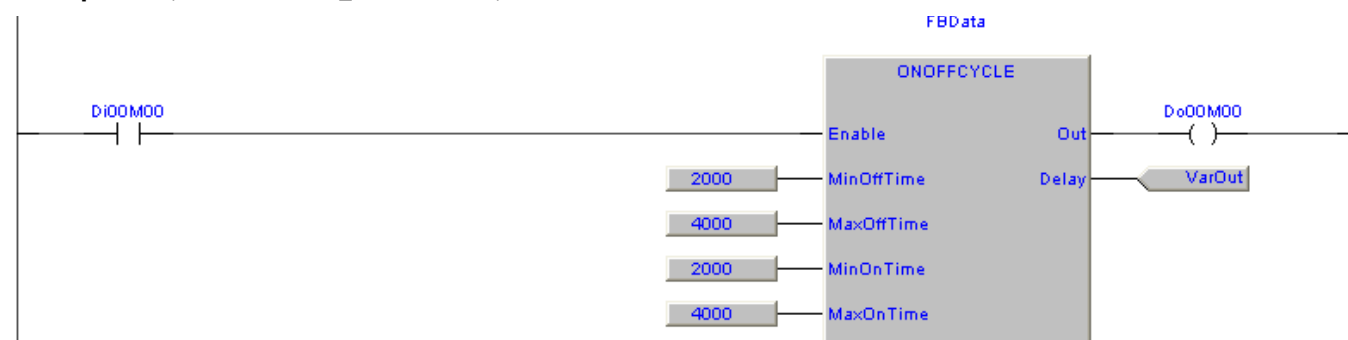
Delay (UDINT) Valore di tempo attualmente in temporizzazione (mS).

Esempi

Viene eseguito il lampeggio della uscita ***Do00M00*** con tempi random variabili tra i 2 ed i 4 secondi.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	ONOFFCYCLE	Auto	No	0	..	ONOFFCYCLE FB data
2	VarOut	UDINT	Auto	No	0	..	Variable output

Esempio LD *(PTP114A100, LD_ONOFFCYCLE)*

Esempio IL *(PTP114A100, IL_ONOFFCYCLE)*

```
CAL FBData (* Call the ONOFFCYCLE function block *)

LD  Di00M00
ST  FBData.Enable (* Transfer the digital input to Enable input *)

LD  2000
ST  FBData.MinOffTime (* Set the minimum off time *)

LD  4000
ST  FBData.MaxOffTime (* Set the maximum off time *)

LD  2000
ST  FBData.MinOnTime (* Set the minimum on time *)

LD  4000
```

```
ST  FBData.MaxOnTime (* Set the maximum on time *)

LD  FBData.Out
ST  Do00M00 (* Copy the Out value to logic output *)

LD  FBData.Delay
ST  VarOut (* The Delay time is copied to variable *)
```

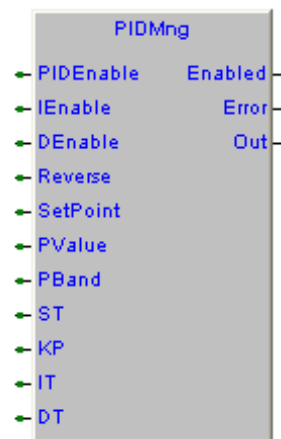
Esempio ST *(PTP114A100, ST_ONOFFCYCLE)*

```
FBData(); (* Call the ONOFFCYCLE function block *)

FBData.Enable:=Di00M00; (* Transfer the digital input to Enable input *)
FBData.MinOffTime:=2000; (* Set the minimum off time *)
FBData.MaxOffTime:=4000; (* Set the maximum off time *)
FBData.MinOnTime:=2000; (* Set the minimum on time *)
FBData.MaxOnTime:=4000; (* Set the maximum on time *)

Do00M00:=FBData.Out; (* Copy the Out value to logic output *)
VarOut:=FBData.Delay; (* The Delay time is copied to variable *)
```

7.13.7 PIDMng, PID management

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione esegue la regolazione PID. E' prevista la possibilità di abilitare singolarmente i vari tipi di azione (***P***)roporzionale (***I***)ntegrativa, (***D***)erivativa.

Un comando di ***Reverse*** permette di invertire il segno del segnale in uscita ***Out***.

PIDEnable (BOOL) Abilitazione regolazione PID, attivando l'ingresso si abilita la regolazione. Disattivando l'ingresso si azzera il valore in uscita ***Out***.

IEnable (BOOL) Abilitazione regolazione integrativa, attivando l'ingresso si abilita la regolazione integrativa.

DEnable (BOOL) Abilitazione regolazione derivativa, attivando l'ingresso si abilita la regolazione derivativa.

Reverse (BOOL) Inversione segno su valore in uscita ***Out***.

SetPoint (REAL) Set point, il valore è espresso nell'unità di misura del processo da controllare.

PValue (REAL) Valore acquisito dal processo, il valore è espresso nell'unità di misura del processo da controllare.

PBand (REAL) Banda proporzionale, questo valore definisce il valore di errore oltre al quale la regolazione viene disabilitata forzando l'uscita ***Out*** al massimo ***±100%***. Il valore è espresso nell'unità di misura del processo da controllare.

ST (REAL) Tempo di scansione, occorre impostare il tempo in cui si desidera vengano eseguite le regolazioni integrativa e derivativa se abilitate, il valore è in ***mS***.

KP (REAL) Costante proporzionale, si ricorda che più è elevato il valore più è pronta è la regolazione con un conseguente aumento del valore di overshot. Il valore è un numero.

IT (REAL) Tempo integrativo, si ricorda che più è elevato il valore meno è veloce la regolazione integrativa a recuperare l'errore. Il valore è espresso in ***Sec***.

DT (REAL) Tempo derivativo, si ricorda che più è elevato il valore più è veloce la regolazione derivativa a recuperare l'errore. Il valore è espresso in ***Sec***.

Enabled (BOOL) Regolazione PID abilitata.

Error (BOOL) Errore nella esecuzione.

OUT (REAL) Valore di correzione in uscita dalla regolazione PID. Questo valore deve essere utilizzato per il comando del processo. Il valore è espresso in ***%***. Il range è compreso tra ***±100%***.

Codici di errore

In caso di errore si attiva l'uscita ***Error***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10012050 Non è stato definito valore di ***ST***.

Esempi

Nell'esempio è gestita una regolazione di temperatura su di un termoriscaldatore. Viene acquisita la sonda di temperatura da una Pt100 e viene gestita una uscita PWM ***Do00M00*** per il comando. Le costanti del loop PID sono allocate in memoria backup e sono mantenute allo spegnimento, inoltre sono accessibili da modbus.

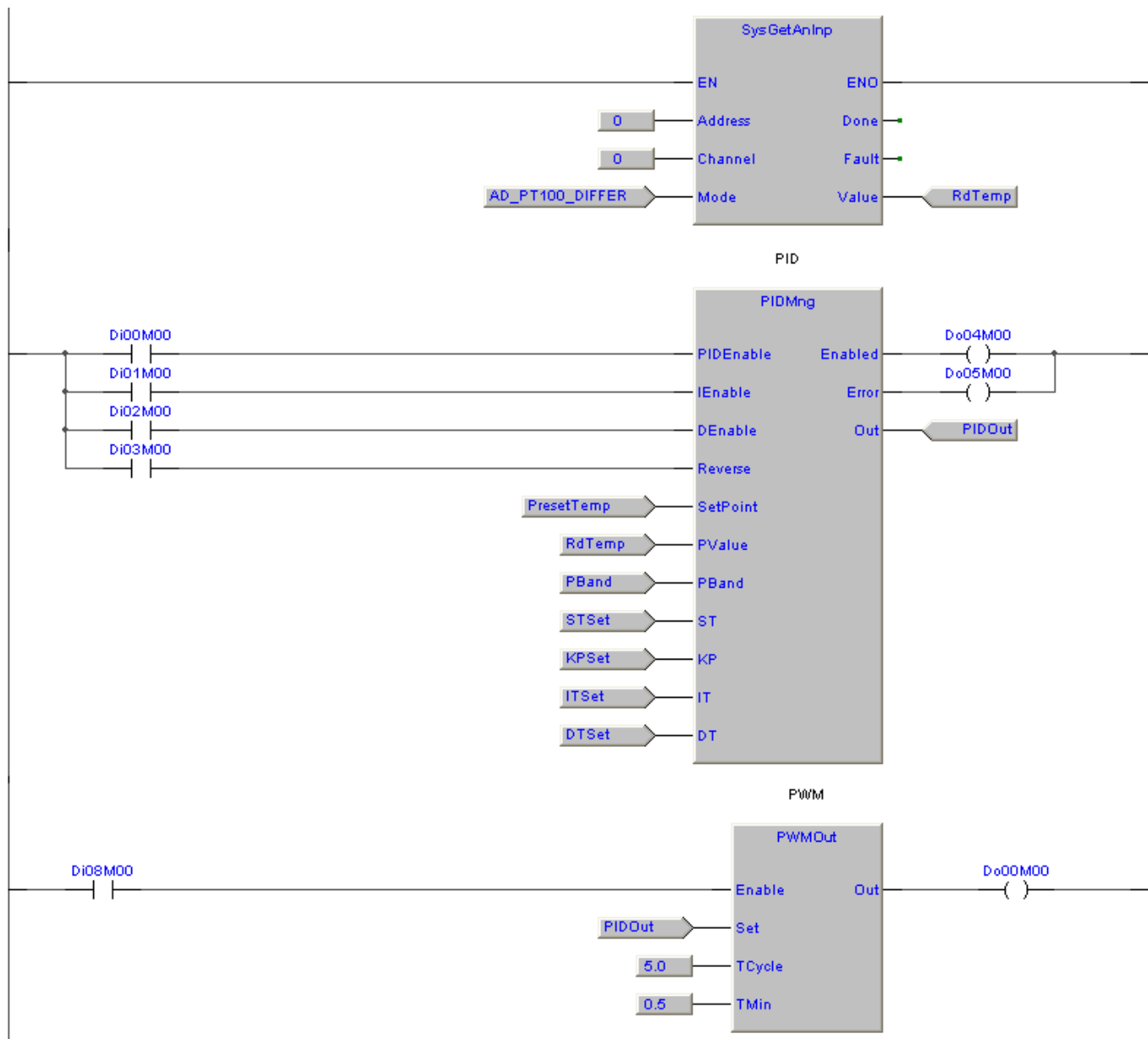
Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RdTemp	REAL	%MD100.0	No	0	..	Read temperature (Degrees)
2	PresetTemp	REAL	%MD100.2048	No	0	..	Set point temperature (Degrees)
3	PBand	REAL	%MD100.2052	No	0	..	Proportional band (Degrees)
4	STSet	REAL	%MD100.2056	No	0	..	Scansion time (mS)
5	KPSet	REAL	%MD100.2060	No	0	..	Proportional coefficent
6	ITSet	REAL	%MD100.2064	No	0	..	Integrative time (S)
7	DTSet	REAL	%MD100.2068	No	0	..	Derivative time (S)
8	TempRead	SysGetAnInp	Auto	No	0	..	Analog input
9	PWM	PWMOut	Auto	No	0	..	PWM management
10	PIDOut	REAL	Auto	No	0	..	PID output value (%)
11	PID	PIDMng	Auto	No	0	..	PID management

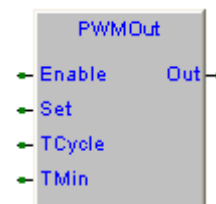
Esempio LD *(PTP114A100, LD_PIDMng)*

SysGetAnInp
EN ENO
0 Address Done
0 Channel Fault
AD_PT100_DIFFER Mode Value RdTemp

PID
PIDMng
Di00M00 PIDEnable Enabled Do04M00
Di01M00 IEnable Error Do05M00
Di02M00 DEnable Out PIDOut
Di03M00 Reverse
PresetTemp SetPoint
RdTemp PValue
PBand PBand
STSet ST
KPSet KP
ITSet IT
DTSet DT

PWM
PWMOut
Di08M00 Enable Out Do00M00
PIDOut Set
5.0 TCycle
0.5 TMin

7.13.8 PWMOut, PWM output management

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione esegue la gestione di una uscita PWM.

Enable (BOOL) Abilitazione gestione uscita PWM, attivando l'ingresso si abilita la gestione. Disattivando l'ingresso si azzera l'uscita ***Out***.

SET (REAL) Valore di set PWM, il valore è espresso in **%**.

TCycle (REAL) Tempo di di ciclo PWM, il valore è espresso in ***S***.

TMin (REAL) Tempo minimo comando uscita ***Out***, il valore è espresso in ***S***.

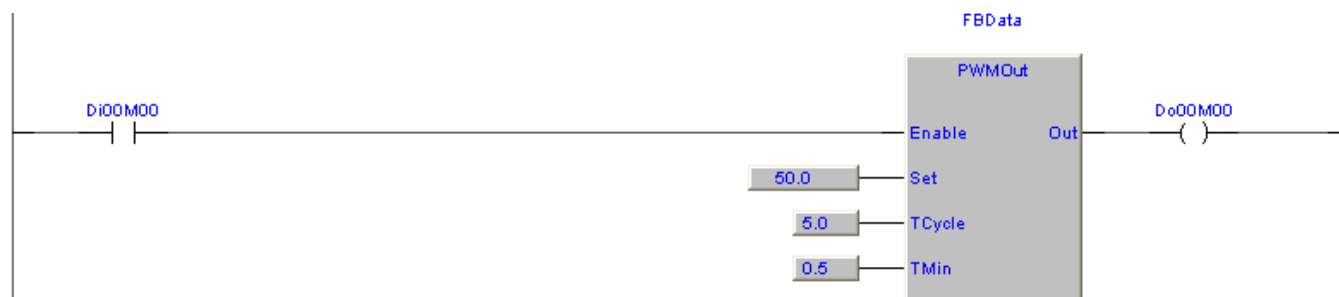
Out (BOOL) Uscita PWM.

Esempi

Nell'esempio è gestita una uscita PWM definendo un tempo di ciclo di 5 secondi con un tempo minimo di 0.5 secondi. Impostando come set point il valore 50% attivando l'ingresso ***Di00M00*** avremo che l'uscita ***Do00M00*** sarà attivata per 2.5 secondi e disattivata per 2,5 secondi.

Definizione variabili

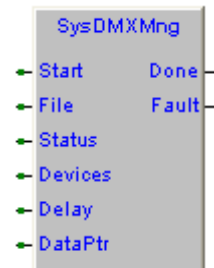
	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	PWMOut	Auto	No	0	..	FB PWMOut data

Esempio LD *(PTP114A100, LD_PWMOut)*

7.13.9 SysDMXMng, DMX management

Type	Library
FB	XTarget_07_0

Questo blocco funzione esegue la gestione del protocollo DMX, questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi ***protezione funzioni e blocchi funzione***. E' comunque possibile utilizzarlo liberamente in modo test per 15 Min.

Attivando l'ingresso di ***Start*** viene inviato sulla porta seriale identificata da ***File*** un frame DMX che inizia con il valore di ***Status*** e segue con il valore di preset dei dispositivi definiti da ***Devices***. Il valore di preset dei vari dispositivi deve essere caricato in un array di dati il cui indirizzo è passato in ***DataPtr***. Mantenendo sempre attivo l'ingresso ***Start*** verranno inviati consecutivamente frames DMX.

Al termine dell'invio del comando DMX si attiverà per un loop di programma l'uscita ***Done***.

Start (BOOL) Comando di invio frame DMX su porta seriale, si resetta automaticamente all'invio del frame.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

Status (USINT) Valore byte di stato inviato nel protocollo DMX prima dei byte di preset dispositivi.

Devices (UINT) Numero di dispositivi connessi al bus DMX.

Delay (UINT) Tempo di pausa trasmissione frames DMX (mSec)

DataPtr (@USINT) Pointer all'array dati valori di preset dispositivi DMX.

Done (BOOL) Attivo per un loop al termine dell'invio frame DMX del comando

Fault (BOOL) Attivo in caso di errore nella gestione.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

9979050 Errore allocazione blocco funzione.

9979060 Terminato spazio memoria rilocabile, non è possibile eseguire l''FB.

9979070 Errore versione blocco funzione.

9979085 FB protetta, terminato tempo funzionamento in modo demo.

9979200 Protocollo DMX non supportato dal dispositivo definito in ***File***.

9979990 Non implementata nel simulatore.

Esempi

Dovendo gestire puntatori a memoria è preferibile utilizzare il blocco funzione all'interno di un programma ST, nell'esempio viene attivato il protocollo DMX sulla porta seriale ***COM1***. Vengono gestiti 5 dispositivi con indirizzi da 1 a 5. Il frame DMX è continuamente inviato ai dispositivi.

Attivando l'ingresso digitale ***Di00M00*** viene impostato il valore 0, su tutti i dispositivi. Attivando l'ingresso digitale ***Di01M00*** su dispositivo 1 viene impostato il valore 10, sul 2 il valore 20, e così via fino al quinto in cui viene impostato il valore 50.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBDMX	SysDMXMng	Auto	No	0	..	FB gestione protocollo DMX
2	DMXData	USINT	Auto	[0..4]	5(0)	..	DMX data
3	DiPls	R_TRIG	Auto	[0..1]	0,0	..	Pulse su ingresso

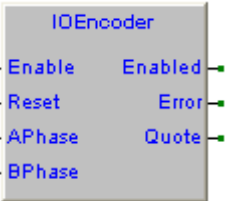
Esempio ST

```
(* -------------------------------------------------------------------------- *)
(* ESEGUO APERTURA PORTA SERIALE                                              *)
(* -------------------------------------------------------------------------- *)
(* Here the COM1 port is opened in read/write. *)

    IF (FBDMX.File = NULL) THEN
        FBDMX.File:=Sysfopen('COM1', 'rw'); (* Port COM1 file pointer *)
    END_IF;

(* -------------------------------------------------------------------------- *)
(* ESEGUO ATTIVAZIONE COMANDI                                                 *)
(* -------------------------------------------------------------------------- *)
(* Attivazione comandi su input Di00M00. *)

    DiPls[0](CLK:=Di00M00);
    IF (DiPls[0].Q) THEN
        DMXData[0]:=0; (* Preset dispositivo con indirizzo 1 *)
        DMXData[1]:=0; (* Preset dispositivo con indirizzo 2 *)
        DMXData[2]:=0; (* Preset dispositivo con indirizzo 3 *)
        DMXData[3]:=0; (* Preset dispositivo con indirizzo 4 *)
        DMXData[4]:=0; (* Preset dispositivo con indirizzo 5 *)
        FBDMX.Start:=TRUE; (* Start *)
    END_IF;

(* Attivazione comandi su input Di01M00. *)

    DiPls[1](CLK:=Di01M00);
    IF (DiPls[1].Q) THEN
        DMXData[0]:=10; (* Preset dispositivo con indirizzo 1 *)
        DMXData[1]:=20; (* Preset dispositivo con indirizzo 2 *)
        DMXData[2]:=30; (* Preset dispositivo con indirizzo 3 *)
        DMXData[3]:=40; (* Preset dispositivo con indirizzo 4 *)
        DMXData[4]:=50; (* Preset dispositivo con indirizzo 5 *)
       FBDMX.Start:=TRUE; (* Start *)
    END_IF;

(* -------------------------------------------------------------------------- *)
(* ESEGUO GESTIONE PROTOCOLLO DMX                                             *)
(* -------------------------------------------------------------------------- *)
(* Gestione protocollo DMX. *)

    FBDMX.Status:=0; (* Status byte *)
    FBDMX.Devices:=5; (* Number of devices *)
    FBDMX.Delay:=0; (* Interframe delay (mSec) *)
    FBDMX.DataPtr:=ADR(DMXData); (* Data array pointer *)
    FBDMX(); (* FB gestione protocollo DMX *)
    IF (FBDMX.Done) THEN FBDMX.Start:=FALSE; END_IF;

(* [End of file] *)
```

7.13.10 IOEncoder, incremental encoder over I/O

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione esegue la lettura di un encoder incrementale connesso agli ingressi logici. Basta appoggiare sui due ingressi ***APhase*** e ***BPhase*** del blocco funzione i due ingressi di acquisizione del canale ***A*** e del canale ***B*** di un encoder incrementale. Il blocco funzione esegue la quadratura dei segnali, il controllo della direzione di rotazione e gestisce il valore di ***Quote*** in uscita.

La quadratura dei segnali esegue la moltiplicazione per 4 delle tacche encoder quindi il valore di ***Quote*** al termine di un giro completo dell'encoder è pari al numero di tacche encoder moltiplicato per 4.

Enable (BOOL) Abilitazione gestione conteggio encoder.

Reset (BOOL) Comando di reset quota encoder. Attivando l'ingresso si azzera il valore di ***Quote***.

APhase (BOOL) Ingresso canale ***A*** encoder.

BPhase (BOOL) Ingresso canale ***B*** encoder.

Enabled (BOOL) Conteggio encoder abilitato.

Error (BOOL) Attivo per un loop su errore acquisizione encoder. Si attiva se la frequenza di ingresso dei segnali encoder è maggiore rispetto al tempo di esecuzione del blocco funzione.

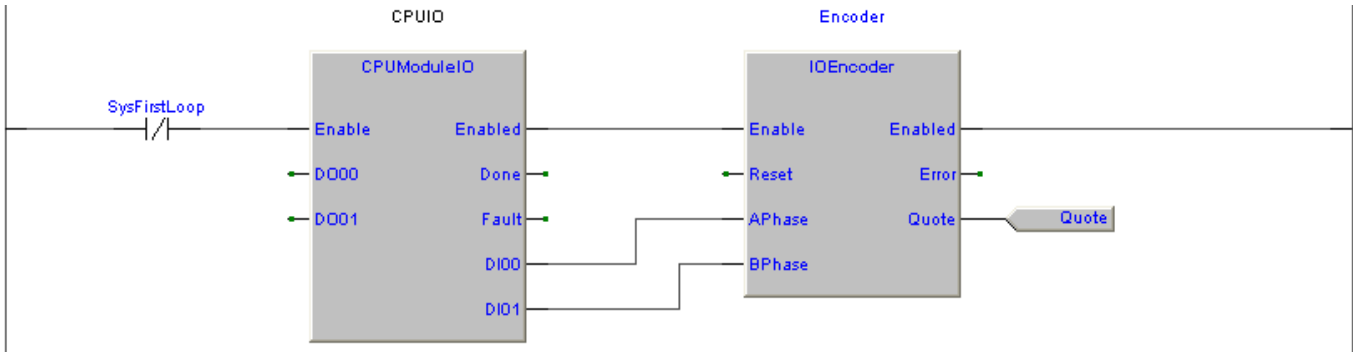
Quote (UDINT) Valore di quota encoder espresso in impulsi. Numero tacche giro encoder moltiplicato per 4.

Esempi

Nell'esempio è gestita l'acquisizione di un encoder incrementale connesso agli ingressi del modulo CPU. Ruotando di un giro l'encoder, il valore di ***Quote*** verrà incrementato se la rotazione è oraria (CW) oppure decrementato se la rotazione è antioraria (CCW), del numero di tacche giro moltiplicato per 4.

Definizione variabili

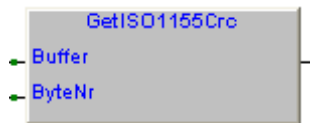
	Name	Type	Address	Array	Init value	Attribute	Description
1	Encoder	IOEncoder	Auto	No	0	..	IOEncoder function block
2	CPUIO	CPUModuleIO	Auto	No	0	..	CPUModuleIO function block
3	Quote	UDINT	Auto	No	0	..	Encoder quote

Esempio LD

7.13.11 GetISO1155Crc, calculate CRC according ISO1155

Type	Library
Function	ePLCUtyLib_C000

Questa funzione esegue il calcolo del CRC ***Cyclic Redundancy Check***, (Controllo Ciclico di Ridondanza) su di un'area dati. Il calcolo è effettuato secondo le specifiche ***ISO 1155***.

Occorre passare alla funzione l'indirizzo del buffer di memoria ***Buffer*** ed il numero di bytes ***ByteNr*** su cui eseguire il calcolo del CRC.

GetISO1155Crc
Buffer
ByteNr

Buffer (@USINT) Indirizzo dell'area di memoria su cui eseguire il calcolo del CRC.

ByteNr (UINT) Numero di bytes su cui eseguire il calcolo del CRC a partire dall'indirizzo definito in ***Buffer***.

Result (UINT) Valore CRC calcolato.

Esempi

Viene calcolato il CRC di una richiesta di lettura del registro 1.8.1 da un contatore di energia elettrica secondo lo standard IEC 62056-2. Il frame di richiesta è ***'<SOH>R1<STX>1.8.1()<ETX><CRC>'***.

Il valore del CRC ritornato in ***CRCValue*** è 16#5A (90 decimale).

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	DataFrame	STRING	Auto	[16]		..	Data frame
2	CRCValue	UINT	Auto	No	0	..	CRC Value

Esempio ST

```
(* ************************************************************************** *)
(* IEC1155 CRC CALCULATION                                                    *)
(* ************************************************************************** *)
(* Register read command '<SOH>R1<STX>1.8.1()<ETX><CRC>'. *)

DataFrame:='$01R1$021.8.1()$03'; (* Data frame *)
CRCValue:=GetISO1155Crc(ADR(DataFrame), 12); (* CRC Value *)
```

7.13.12 IODataExchange, exchange data by using logic I/O

Type	Library
FB	ePLCUtyLib_C000

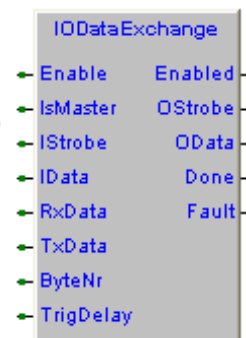
IODataExchange
Enable — Enabled
IsMaster — OStrobe
IStrobe — OData
IData — Done
RxData — Fault
TxData
ByteNr
TrigDelay

Questo blocco funzione permette lo scambio di dati tra due sistemi, uno master ed uno slave, utilizzando una connessione tramite I/O logici. Sono utilizzati due ingressi e due uscite digitali per ogni sistema, è possibile definire il numero di bytes di dati da scambiarsi.

Occorre connettere l'uscita digitale ***OStrobe*** di un sistema con l'ingresso digitale ***IStrobe*** dell'altro sistema e l'uscita ***OData*** con l'ingresso digitale ***IData*** dell'altro.

Il trasferimento dati è bidirezionale, i dati presenti nel buffer ***TxData*** di un sistema sono trasferiti nel buffer ***RxData*** dell'altro sistema e viceversa, per il numero di bytes definito in ***ByteNr***. La comunicazione è verificata mediante l'invio di un CRC secondo lo standard ISO 1155.

Ad ogni ***fine*** trasferimento dati si attiva per un loop l'uscita ***Done***, sulla sua attivazione occorre provvedere a trasferire i dati da trasmettere nel buffer di trasmissione e leggere i dati dal buffer di ricezione.

In caso di ***errore*** nella comunicazione si attiva per un loop l'uscita ***Fault***, ed i due sistemi si risincronizzano per riprendere una nuova trasmissione.

Enable (BOOL) Abilitazione gestione comunicazione.

IsMaster (BOOL) ***TRUE:*** Modo master, ***FALSE:*** Modo slave.

IStrobe (BOOL) Occorre appoggiare l'ingresso digitale di strobe.

IData (BOOL) Occorre appoggiare l'ingresso digitale di dato.

RxData (UDINT) Indirizzo buffer dati ricevuti.

TxData (UDINT) Indirizzo buffer dati da trasmettere.

ByteNr (USINT) Numero bytes da scambiare con altro sistema (Da 1 a 30).

TrigDelay (UINT) Tempo attesa tra uscita dato ***OData*** ed uscita strobe ***OStrobe*** (Da 0 a 30 mS).

OStrobe (BOOL) Da appoggiare su uscita digitale di strobe.

OData (BOOL) Da appoggiare su uscita digitale di dato.

Done (BOOL) Si attiva per un loop al termine dello scambio dati.

Fault (BOOL) Si attiva per un loop in caso di errore su scambio dati.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10011080 Errore definizione valore ***ByteNr***.

10011082 Errore definizione valore ***TrigDelay***.

10011100~1 Timeout attesa attivazione segnale ***IStrobe***.

10011110~1 Timeout attesa disattivazione segnale ***IStrobe***.

10011200~1 Errore CRC dati ricevuti.

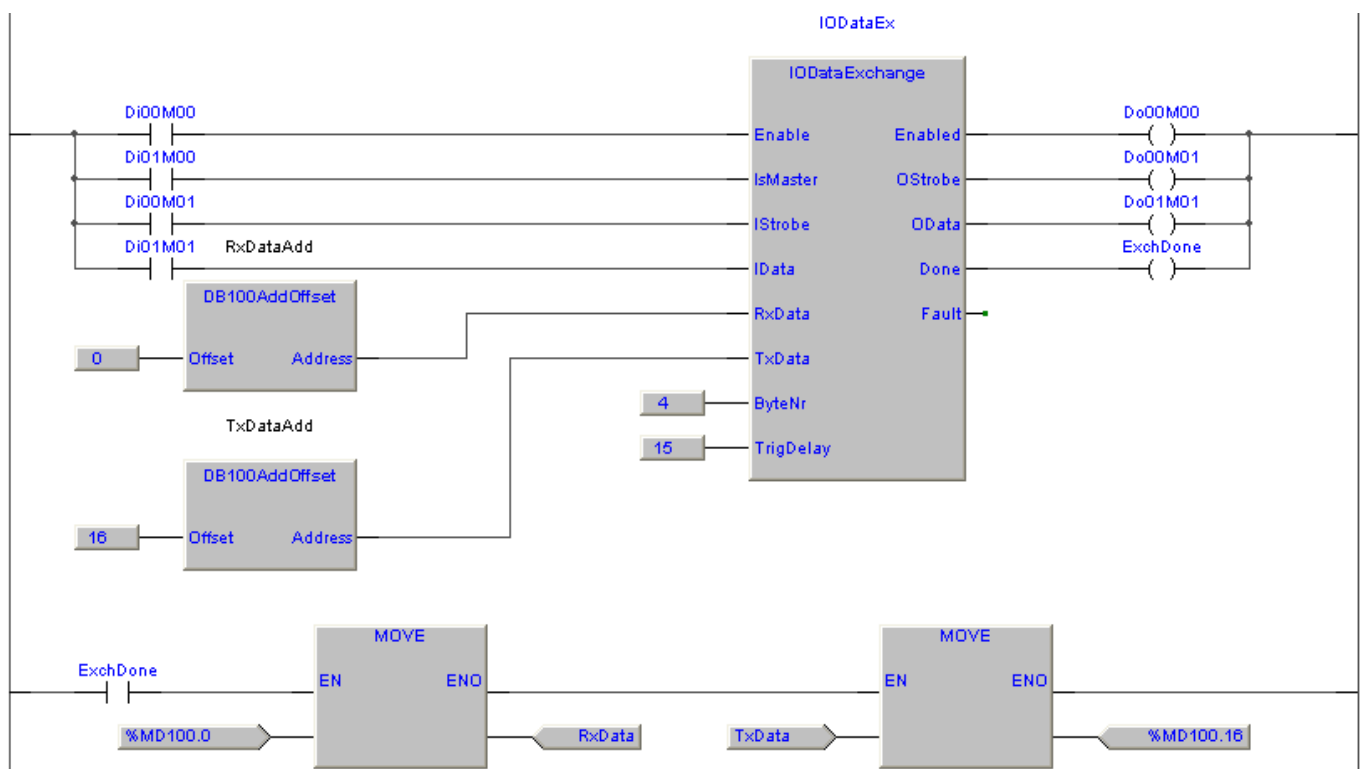
Esempi

Utilizzando due sistemi attivando modo master (***Di01M00*** attivo) su di un sistema e modo slave (***Di01M00*** disattivo) sull'altro, è possibile eseguire lo scambio di 4 bytes di memoria tra i sistemi. I 4 bytes allocati ad indirizzo ***MD100.0*** di un sistema saranno trasferiti su 4 bytes allocati ad indirizzo ***MD100.16*** dell'altro sistema.

Al termine del trasferimento, i dati ricevuti dalla memoria ***MD100.0*** sono trasferiti nella variabile ***RxData***, mentre la variabile ***TxData*** è trasferita in memoria ***MD100.16***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	ExchDone	BOOL	Auto	No	FALSE	..	Data exchange done
2	RxData	UDINT	Auto	No	0	..	Received data
3	TxData	UDINT	Auto	No	0	..	Transmit data
4	RxDataAdd	DB100AddOffset	Auto	No	0	..	RxData address
5	TxDataAdd	DB100AddOffset	Auto	No	0	..	TxData address
6	IODataEx	IODataExchange	Auto	No	0	..	FB IODataExchange data

Esempio LD *(Ptp121A000)*

7.13.13 Average, value average

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione esegue la media su di un valore. L'azione di media è definita da un parametro ***Coefficient***, maggiore è il valore del parametro e maggiore sarà l'azione di media sul valore in uscita ***Average***.

Value (REAL) Valore su cui effettuare l'azione di media

Coefficient (REAL) Valore del coefficiente di media.

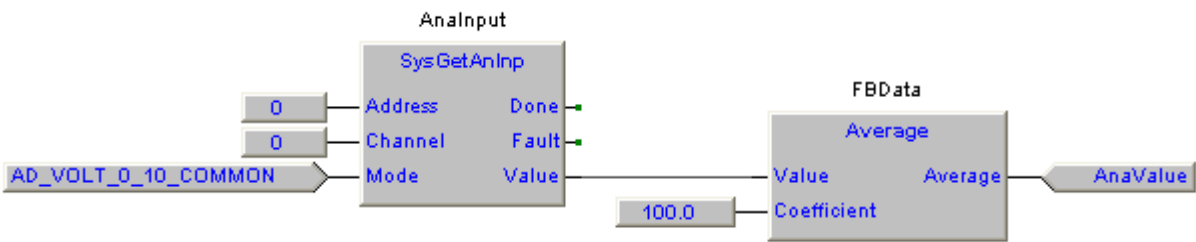
Average (REAL) Valore mediato in uscita.

Esempi

Viene eseguita una acquisizione analogica dall'ingresso ***0*** del modulo con indirizzo ***0***, in modo 0÷10 volt. Il valore acquisito viene mediato e poi trasferito nella variabile ***AnaValue***.

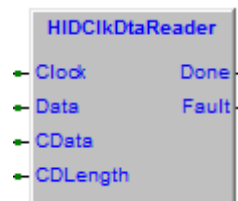
Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	FBData	Average	Auto	No	0	..	FB average data
2	AnaInput	SysGetAnInp	Auto	No	0	..	FB Analog input data
3	AnaValue	REAL	Auto	No	0	..	Analog input value

Esempio FBD *(PTP114A500, FBD_Average)*

7.13.14 HIDClkDtaReader, HID RFID clock/data reader

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione esegue l'acquisizione di un lettore HID tipo clock e dato. La gestione del lettore deve essere eseguita almeno ogni 400 uSec, quindi si consiglia di utilizzarlo in un programma eseguito nella task fast, definendo il tempo di esecuzione della task a 400 uS tramite la funzione ***SysSetTaskLpTime***.

Il blocco funzione può acquisire tags RFID con codice di lunghezza fino ad 8 bytes, il codice letto viene trasferito nel buffer indirizzato da ***CData***, occorre definire la lunghezza del codice in ***CDLength***.

Clock (BOOL) Ingresso di clock del lettore RFID.

Data (BOOL) Ingresso di dato del lettore RFID.

CData (@BYTE) Indirizzo buffer codice letto dal tag RFID.

CDLength (USINT) Numero bytes codice letto dal tag RFID.

Done (BOOL) Attivo per un loop al termine della lettura tag RFID.

Fault (BOOL) Attivo per un loop su errore lettura tag RFID.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10039050 Errore definizione valore ***CDLength***.

10039060 Errore esecuzione.

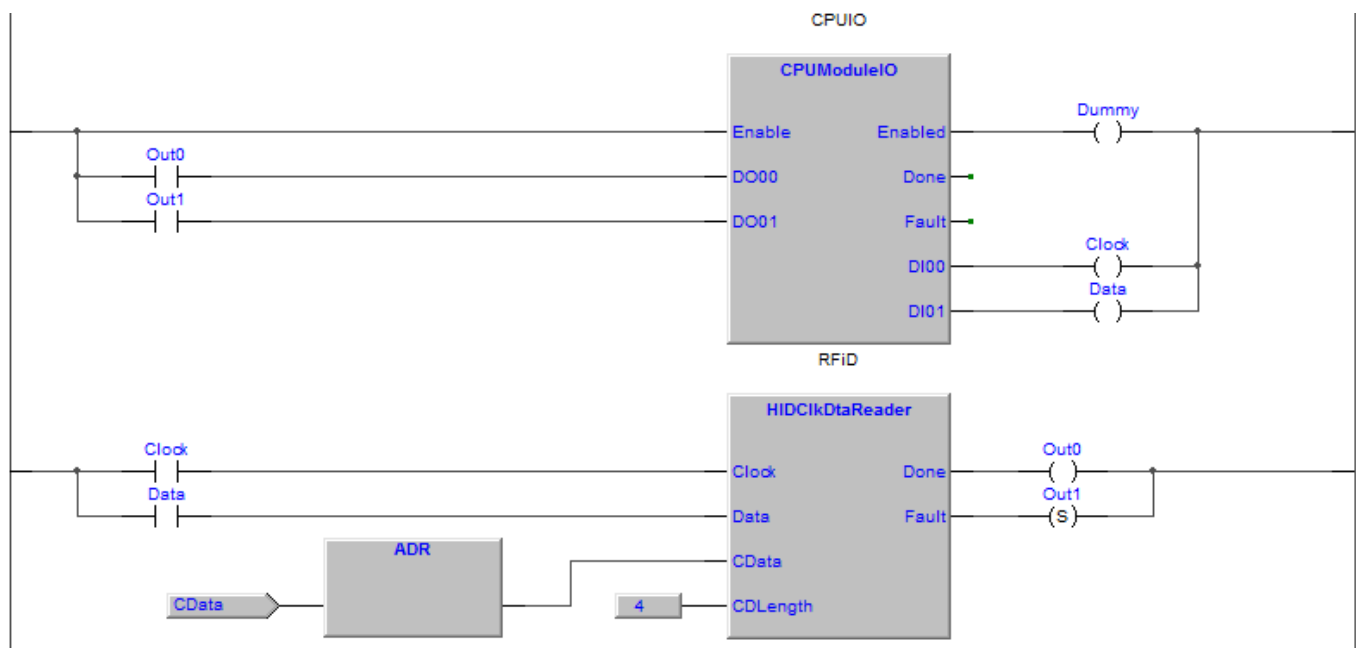
10039100~2 Errore lettura dati tag RFID.

Esempi

Viene gestita la lettura di un reader della HID clock e dato connesso agli ingressi digitali del modulo CPU. Ricordo che il programma deve essere eseguito almeno ogni 400 uS quindi si consiglia di eseguilo nella task Fast impostando il tempo di esecuzione a 400 uS. Il valore di codice letto dal TAG RFID è trasferito nell'array ***CData***.

Definizione variabili

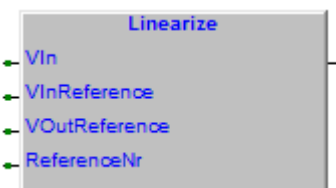
	Name	Type	Address	Array	Init value	Attribute	Description
1	Clock	BOOL	Auto	No	FALSE	..	Clock input signal
2	Data	BOOL	Auto	No	FALSE	..	Data input signal
3	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
4	Out0	BOOL	Auto	No	FALSE	..	Output 0 on CPU module
5	Out1	BOOL	Auto	No	FALSE	..	Output 1 on CPU module
6	CData	BYTE	Auto	[0..7]	8(0)	..	Card data
7	CPUIO	CPUModuleIO	Auto	No	0	..	CPUModuleIO function block
8	RFiD	HIDClkDtaReader	Auto	No	0	..	RFID read

Esempio LD *(PTP114A610, LD_HIDCardRead)*

7.13.15 Linearize, linearize a non linear value

Type	Library
Function	ePLCUtyLib_C000

Linearize
VIn
VInReference
VOutReference
ReferenceNr

Questa funzione esegue la linearizzazione di un valore. Occorre fornire alla funzione l'indirizzo dell'array definizione valore in ingresso da linearizzare **VinReference**, l'indirizzo dell'array definizione del corrispondente valore in uscita linearizzato **VOutReference** ed il numero di valori presenti nell'array **ReferenceNr**. E' importante definire i dati negli array in modo crescente, quindi a partire dal valore più piccolo.

La funzione esegue la ricerca nell'array **VinReference** del valore immediatamente superiore a **Vin** interpolando linearmente tra il valore trovato e quello precedente, calcolando il valore in uscita in base ai due valori presenti nelle stesse posizioni dell'array **VOutReference**.

VIn (REAL) Valore su cui effettuare la linearizzazione

VInReference (@REAL) Puntatore all'array definizione valore in ingresso da linearizzare.

VOutReference (@REAL) Puntatore all'array definizione valore in uscita linearizzato.

ReferenceNr (USINT) Numero di valori negli arrays di definizione linearizzazione.

(REAL) Valore linearizzato in uscita.

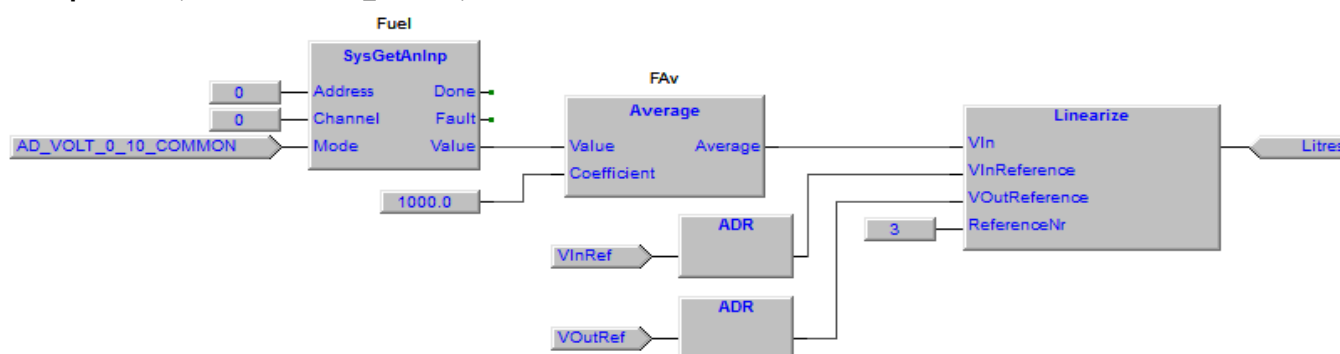
Esempi

Viene eseguita tramite una acquisizione analogica la lettura di un livello di un serbatoio di forma irregolare, tramite la funzione viene ritornato il valore dei litri presenti nel serbatoio. Il valore di livello viene mediato per avere una lettura piu regolare.

Ad 1 volt corrispondono 1000 litri, a 2 volt corrispondono 4000 litri, a 3 volt corrispondono 6000 litri.

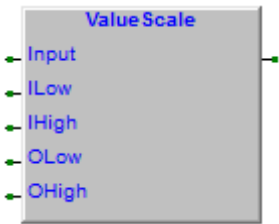
Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fuel	SysGetAnInp	Auto	No	0	..	Analog acquisition
2	Litres	REAL	Auto	No	0	..	Fuel quantity (Litres)
3	VInRef	REAL	Auto	[0..15]	1,2,3,13(0)	..	Level voltage input
4	VOutRef	REAL	Auto	[0..15]	1000,4000,6000,13(0)	..	Litres value
5	FAv	Average	Auto	No	0	..	Level average

Esempio FBD *(PTP114A610, FBD_Linearize)*

7.13.16 ValueScale, scales a value

Type	Library
Function	ePLCUtyLib_C000

Questa funzione esegue la messa in scala di un valore. E' possibile definire il limite minimo e massimo del valore in ingresso da scalare (***ILow***, ***IHigh***) ed il valore minimo e massimo del valore in uscita scalato (***OLow***, ***OHigh***).

L'utilizzo di questa funzione è particolarmente indicato per convertire valori acquisiti da sensori in corrente (4-20 mA) nella corrispondente unità di misura rilevata dal trasduttore.

Input (REAL) Valore in ingresso da scalare.

ILow (REAL) Limite minimo del valore in ingresso da scalare.

IHigh (REAL) Limite massimo del valore in ingresso da scalare.

OLow (REAL) Limite minimo del valore in uscita scalato.

OHigh (REAL) Limite massimo del valore in uscita scalato.

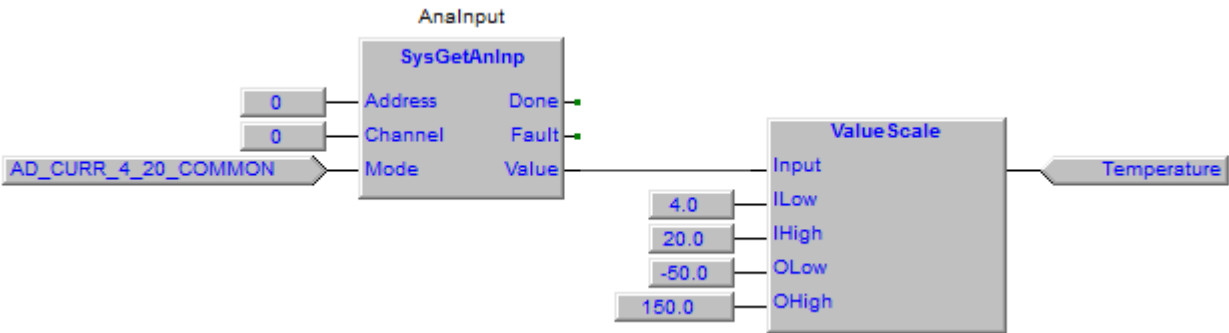
(REAL) Valore scalato in uscita.

Esempi

Viene eseguita una acquisizione analogica di un sensore di temperatura con uscita in corrente 4-20 mA, il sensore ha come valore in uscita la corrente di 4 mA a -50 °C e la corrente di 20 mA a 150 °C. Utilizzando la funzione si esegue direttamente la conversione in gradi centigradi del valore acquisito.

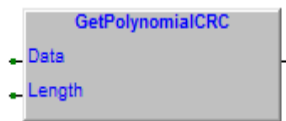
Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	AnaInput	SysGetAnInp	Auto	No	0	..	FB Analog input data
2	Temperature	REAL	Auto	No	0	..	Temperature value (°C)

Esempio FBD *(PTP114A610, FBD_Scale)*

7.13.17 GetPolynomialCRC, polynomial CRC calculation

Type	Library
Function	ePLCUtyLib_C000

Questa funzione esegue il calcolo del CRC su di un array di dati in accordo allo standard CCITT.

In ***Data*** occorre indicare l'indirizzo dell'array di dati su cui calcolare il CRC, ***Length*** indica la lunghezza dell'array in byte. In ***CRC*** è ritornato il valore del CRC calcolato.

Data (@USINT) Indirizzo array dati su cui eseguire il calcolo del CRC.

Length (USINT) Lunghezza array in bytes.

Result (WORD) Valore CRC calcolato.

Esempi

Viene eseguito il calcolo del CRC su di un array di dati. Supponendo di avere un array di 8 bytes allocato in DB100 all'indirizzo 16 che contiene i valori 16#04, 16#3B, 16#1B, 16#00, 16#00, 16#00, 16#00, 16#00 , il CRC calcolato sarà 16#5AF0.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	CRCValue	WORD	Auto	No		..	CRC calculated value
2	BData	BYTE	Auto	[0..7]		..	Byte data

Esempio LD *(PTP114A620, ST_GetPolynomialCRC)*

```
BData[0]:=16#04;
BData[1]:=16#3B;
BData[2]:=16#1B;
BData[3]:=16#00;
BData[4]:=16#00;
BData[5]:=16#00;
BData[6]:=16#00;
BData[7]:=16#00;
CRCValue:=GetPolynomialCRC(ADR(BData), 8);
```

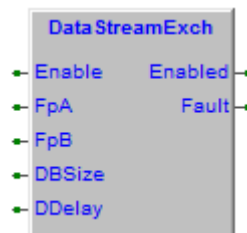
7.13.18 DataStreamExch, exchanges data between two I/O streams

Type	Library
Function	ePLCUtyLib_C000

Questo blocco funzione esegue lo scambio data tra due stream di I/O. Definendo uno stream come socket TCP e l'altro come seriale è possibile realizzare un convertitore Ethernet/Seriale.

In ***FpA*** ed ***FpB*** occorre indicare il file pointer in uscita dalla funzione ***Sysfopen*** di apertura dello stream di I/O. Il FB riceve i dati da uno stream li bufferizza in un buffer di dimensione ***DBSize*** e quando non riceve più dati per il tempo definito in ***DDelay*** li invia sull'altro stream.

Se sono ricevuti più dati della dimensione definita del buffer i dati sono immediatamente inviati sull'altro stream e viene generato un errore.

DataStreamExch
Enable — Enabled
FpA — Fault
FpB
DBSize
DDelay

Enable (BOOL) Abilitazione blocco funzione.

FpA (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

FpB (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

DBSize (UDINT) Dimensione buffer dati (Bytes).

DDelay (UDINT) Tempo pausa attesa trasmissione dati (mS).

Enabled (BOOL) Blocco funzione abilitato.

Fault (BOOL) Attivo per un loop di programma in caso di errore.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10048100 Errore allocazione buffer dati.

10048200 Buffer pieno su ricezione dati da stream (A).

10048250 Buffer pieno su ricezione dati da stream (B).

Esempi

Semplice convertitore Ethernet/Seriale, accetta connessione TCP su porta 1000 e utilizza la porta seriale COM0.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	SMode	SetSMode	Auto	No		..	Set serial mode
2	SktLsn	SysSktListen	Auto	No		..	FB Socket listen
3	SktDone	BOOL	Auto	No		..	Socket done
4	SktConnect	BOOL	Auto	No		..	Socket connect
5	Fp	FILEP	Auto	[0..1]		..	File pointer
6	Dx	DataStreamExch	Auto	No		..	FB data stream exchange

Esempio LD *(PTP114A630, LD_DataStreamExch)*

SysFirstLoop
Sysfopen
EN ENO
'TCPSKT' FName
'rw' Mode
Fp[0]

SktLsn
SysSktListen
SysFirstLoop
Enable Done SktDone
Fp[0] File Fault
MyIP Connect SktConnect
1000 MyPort ClientIP
120 LifeTm ClientPort
100 FlushTm
250 RxSize
250 TxSize

SysFirstLoop
Sysfopen
EN ENO
'COM0' FName
'rw' Mode
Fp[1]

Dx
DataStreamExchange
SktConnect
Enable Enabled
Fp[0] FpA
Fp[1] FpB
300 DBSize
30 DDelay

7.14 Protocollo DLMS, o IEC 62056-21

Con il nuovo protocollo standard DLMS (Device Language Message Specification), la comunicazione con i sistemi di metering viene immensamente semplificata. Questo protocollo ha una struttura orientata agli ***Oggetti*** che rende possibile leggere con la stessa identica modalità dati applicativi provenienti da contatori di diversi costruttori.

Lo standard IEC 61107 o IEC 62056-21 è uno standard internazionale che descrive il protocollo DLMS per la lettura da parte di un computer dei dati da contatori tariffari di energia elettrica, acqua e gas.

Il protocollo prevede una fase di ***Sign-On*** con il contatore durante la quale occorre fornire un codice di accesso (Solitamente il numero di serie del contatore), ed il contatore fornisce una password in uscita che può essere utilizzata per criptare i dati.

Terminata questa fase è possibile richiedere al contatore il valore dei suoi registri utilizzando i codici di identificazione ***OBIS*** a 5 caratteri (IEC 62056-61).

Per la famiglia SlimLine abbiamo sviluppato un apposito blocco funzione che automatizza tutte le operazioni, occorre passare il numero di serie del contatore ed il codice OBIS del registro da leggere. Il blocco funzione esegue il Sign-On sul contatore e ritorna il valore del registro indicato.

Interfaccia con il contatore

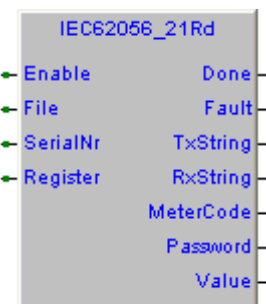
Per interfacciarsi con il contatore è possibile utilizzare un apposito accoppiatore ottico che si appoggia alla finestra di lettura del contatore e si connette ad una delle porte seriali dello SlimLine.

Oppure nel caso di contatori predisposti con l'uscita RS485 è possibile connettersi direttamente alla porta RS485 dello SlimLine.

7.14.1 IEC62056_21Rd, IEC62056-21 protocol read

Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione esegue la gestione della lettura di registri da sistemi di metering utilizzando il protocollo IEC62056-21.Questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi **protezione funzioni e blocchi funzione**. E' comunque possibile utilizzarlo liberamente in modo test per 30 Min.

Occorre in ***SerialNr*** definire il numero seriale del contatore (Utilizzato come chiave di accesso alla lettura) e in ***Register*** l'indirizzo del registro da leggere secondo la normativa OBIS. Se la lettura ha esito positivo viene attivata l'uscita ***Done*** e le variabili ***MeterCode***, ***Password*** e ***Value*** sono valorizzate con i dati letti dal contatore.

Enable (BOOL) Attivando l'ingresso viene gestita la lettura del contatore.

File (FILEP) Pointer al file della risorsa così come ritornato dalla funzione ***Sysfopen***.

SerialNr (STRING[16]) Numero di serie del contatore, viene utilizzato come chiave di accesso.

Register (STRING[16]) Indirizzo registro da leggere secondo la codifica OBIS.

Done (BOOL) Viene attivato per un loop al termine della acquisizione.

Fault (BOOL) Viene attivato per un loop in caso di errore nella sequenza di acquisizione.

TxString (STRING[32]) Contiene la stringa di comando inviata al contatore, può essere utilizzato in debug per verificare i comandi inviati.

RxString (STRING[32]) Contiene la stringa di risposta ritornata dal contatore, può essere utilizzato in debug per verificare le risposte ricevute.

MeterCode (STRING[32]) Contiene stringa con il codice del contatore acquisita durante fase di acceso (Sign-on).

Password (UDINT) Contiene valore password acquisita dal contatore durante fase di acceso (Sign-on).

Value (STRING[32]) Contiene stringa con valore registro richiesto acquisita dal contatore.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10016010 Valore di ***File*** non definito.

10016020 FB protetta, terminato tempo funzionamento in modo demo.

10016050 Timeout esecuzione.

10016070 Errore case gestione.

10016100~1 Errore ricezione tipo da contatore.

10016110~2 Errore ricezione password da contatore.

10016120~2 Errore ricezione valore parametro da contatore.

10016200 Overflow ricezione stringa da contatore.

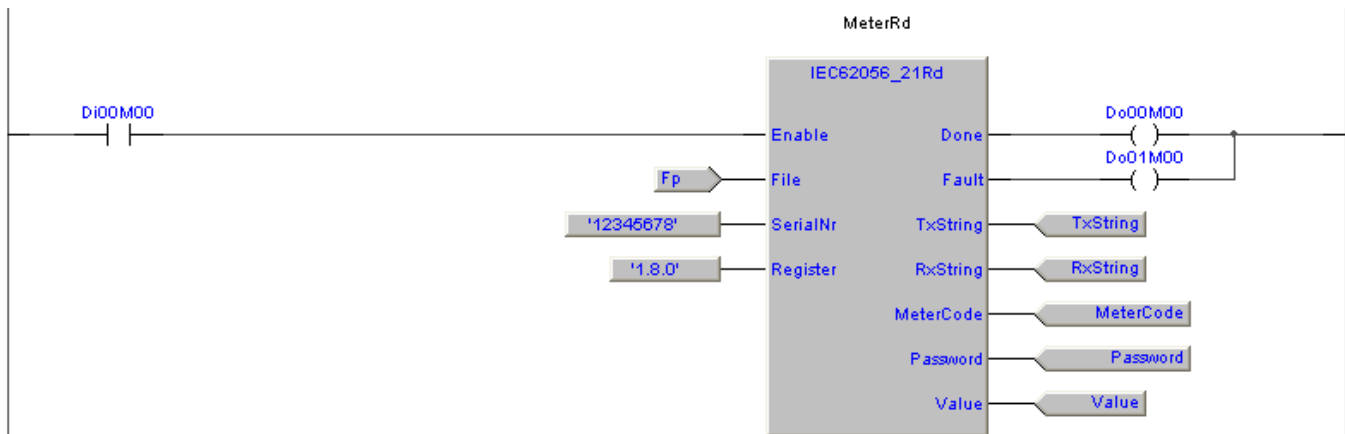
Esempi

E' disponibile un programma di esempio Ptp122*000 che esegue la lettura da un contatore di energia di 3 registri. A titolo indicativo diamo un esempio di massima per la lettura di un registro.

Su attivazione dell'ingresso digitale ***Di00M00*** viene eseguita la lettura del registro ***1.8.0*** (Potenza in Kw). Se la lettura ha esito positivo si attiva per un loop l'uscita digitale ***Di00M00***. Le variabili ***MeterCode***, ***Password*** e ***Value***, saranno valorizzate con i valori letti dal contatore.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	MeterRd	IEC62056_21	Auto	No	0	..	IEC62056_21Rd FB data
3	TxString	STRING	Auto	[32]		..	Tx data string
4	RxString	STRING	Auto	[32]		..	Rx data string
5	MeterCode	STRING	Auto	[32]		..	Meter code
6	Password	UDINT	Auto	No	0	..	Meter password
7	Value	STRING	Auto	[32]		..	Variable read value

Esempio LD *(PTP114A000, LD_IEC62056_21Rd)*

7.15 Funzioni ed FB gestione modem (eModemLib_E000)

Attenzione! Per utilizzare la libreria occorre importarla nel proprio progetto. Vedere capitolo relativo all'import delle librerie.

Le funzioni ed i blocchi funzione per la gestione del modem utilizzano un modem GSM connesso ad un terminale di I/O del sistema (Tipicamente è utilizzata una porta seriale). Nel modem deve essere inserita una tessera SIM ***non protetta dal codice PIN***.

Nella descrizioni successive si fa riferimento alle seguenti definizioni generali.

Numero di telefono

Il numero di telefono consiste in una stringa lunga da 10 a 16 caratteri numerici conforme al seguente formato:

Prefisso internazionale senza lo zero davanti (es. +39 per Italia, +49 per Germania, +44 per Gran Bretagna ecc.)
Codice dell'operatore mobile (es. 338, 320, 347, ecc.)
Numero di telefono (es. 7589951)

Esempio: +393337589951,+3933812345,+49172123456

Messaggio SMS

Un messaggio SMS può essere lungo fino a 160 caratteri alfanumerici facenti parte del seguente set:

A…Z, a…z, 0…9, Spazio bianco, sono da evitare tutti gli altri caratteri.

7.15.1 ModemCore_v3, modem core management

Type	Library
FB	eModemLib_E000

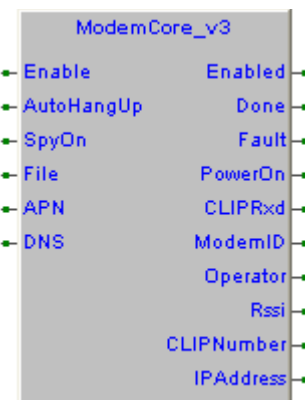
ModemCore_v3
Enable — Enabled
AutoHangUp — Done
SpyOn — Fault
File — PowerOn
APN — CLIPRxd
DNS — ModemID
Operator
Rssi
CLIPNumber
IPAddress

Questo blocco funzione gestisce un modem connesso al dispositivo di I/O definito in ***File***, questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi ***protezione funzioni e blocchi funzione***. E' comunque possibile utilizzarlo liberamente in modo test per 30 Min.

L'FB gestisce il dialogo con il modem, ne esegue l'inizializzazione e ne controlla lo stato, controlla se il modem è connesso alla rete GSM e ritorna l'operatore di rete ***Operator*** ed il livello del segnale ***Rssi***. Nel caso in cui il modem si sganci dalla rete l'FB provvede al suo riaggancio automatico. L'FB ritorna un ***ModemID*** che deve essere passato alle FB collegate (Esempio invio SMS, ricezione SMS, ecc.).L'ingresso ***SpyOn*** se attivo permette di spiare il funzionamento della FB.

L'uscita ***Done*** si attiva se il modem è correttamente inizializzato, mentre l'uscita ***Fault*** si attiva per un loop di programma in caso di errori di gestione.

E' previsto un comando ***PowerOn*** per la gestione della alimentazione del modem, in questo modo l'FB può spegnere e riaccendere il modem in caso riscontri una non funzionalità dello stesso.

Su ricezione chiamata telefonica viene rilevato il CLIP del chiamante che è ritornato in uscita ***CLIPNumber***, contemporaneamente ad ogni squillo del telefono si attiva per un loop di programma l'uscita ***CLIPRxd***. Su connessione GPRS viene ritornato l'indirizzo IP assegnato dal gestore alla connessione ***IPAddress***.

Enable (BOOL) Abilitazione blocco funzione, attivandolo viene gestito il modem.

AutoHangUp (BOOL) Abilita l'HangUp automatico alla ricezione di una chiamata telefonica, viene comunque ritornato il CLIP.

SpyOn (BOOL) Se attivo permette di spiarne il funzionamento.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

APN (STING[32]) Definizione APN per connessione GPRS.

DNS (STING[16]) Definizione indirizzo IP server DNS.

Enabled (BOOL) Blocco funzione abilitato.

Done (BOOL) Modem correttamente inizializzato e funzionante.

Fault (BOOL) Attivo per un loop di programma se errore gestione modem.

PowerOn (BOOL) Comando di gestione uscita alimentazione modem.

CLIPRxd (BOOL) Attivo per un loop di programma ad ogni ricezione CLIP (Tipicamente ad ogni RING del modem).

ModemID (UDINT) ID modem da passare alle FB collegate (Esempio ***ModemSMSSend***, ***ModemSMSReceive***, ecc.).

Operator (STRING[16]) Contiene la stringa con il nome dell'operatore telefonico.

Rssi (USINT) Valore potenza segnale radio.

CLIPNumber (STRING[16]) Contiene la stringa con il numero di CLIP ricevuto.

IPAddress (STING[16]) Indirizzo IP assegnato dal gestore su connessione GPRS.

Trigger di spy

Se ***SpyOn*** attivo viene eseguita la funzione ***SysSpyData*** che permette di spiare il funzionamento della FB. Sono previsti vari livelli di triggers.

TFlags	Descrizione
16#00000001	'Rx' Ricezione stringa da modem
16#00000002	'R>' Stringa ricevuta da modem troppo lunga
16#00000004	'Tx' Trasmissione stringa verso modem
16#00000010	'Mr' Ricezione messaggio SMS
16#01000000	'--' Etichette di segnalazione operazioni modem

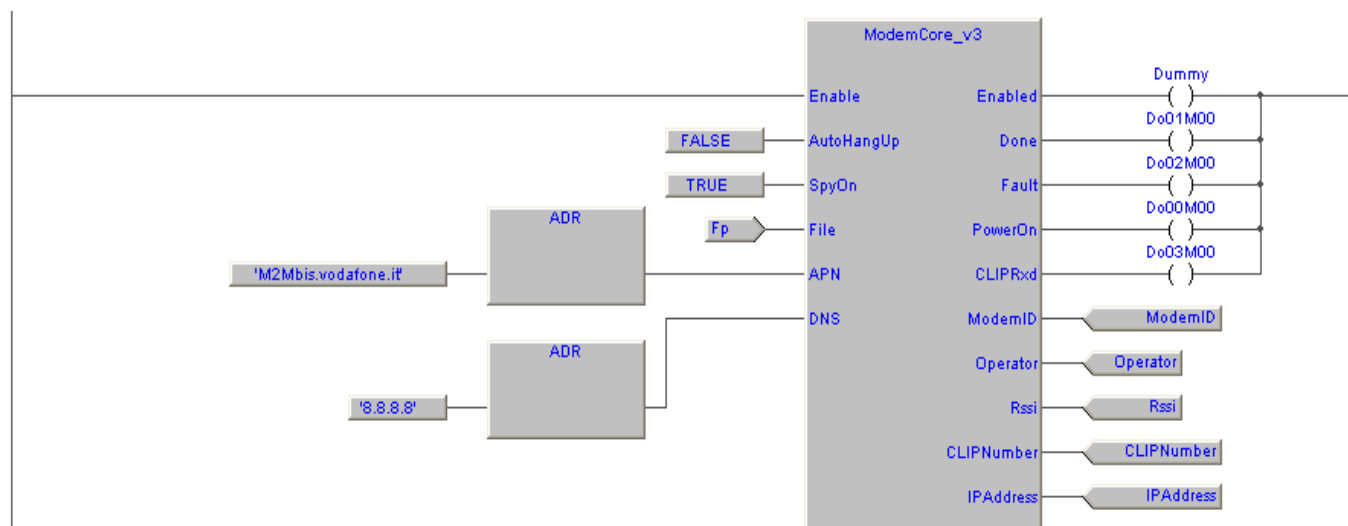
Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

Codice	Descrizione
10002010	Valore di ***File*** non definito.
10002020	FB protetta, terminato tempo funzionamento in modo demo.
10002050	Timeout esecuzione.
10002080	Errore case gestione.
10002100	FB eseguita in task diversa da quella di background.
10002200~3	Errore ricezione CLIP.
10002300~4	Errore nelle sequenze power on del modem.
10002400~10	Errore nelle sequenze di controllo del modem.
10002500~7	Errore nella ricezione del messaggio SMS.

Esempi

Nell'esempio è gestito un modem connesso al terminale di I/O definito nella variabile ***Fp***, per la definizione delle variabili e per una migliore comprensione del funzionamento si rimanda agli esempi successivi.

Esempio LD

7.15.2 ModemSMSReceive, receive a SMS message

Type	Library
FB	eModemLib_E000

Questo blocco funzione esegue la ricezione di un messaggio SMS, si collega al blocco funzione di gestione modem ***ModemCore***, occorre passare il ***ModemID*** in uscita dal blocco funzione di gestione modem.

Alla ricezione di un messaggio SMS si attiva per un loop di programma l'uscita ***Done***, sull'uscita ***SMSText*** viene ritornato il messaggio ricevuto, all'uscita ***CLIPNumber*** della FB ***ModemCore*** è ritornato il numero di telefono da cui il messaggio è stato ricevuto. Il testo del messaggio ricevuto rimane presente in uscita sino alla ricezione di un altro messaggio.

Enable (BOOL) Abilita la ricezione dei messaggi SMS.

ModemID (UDINT) ID modem fornito in uscita dalla ***ModemCore***.

Done (BOOL) Attivo per un loop se ricevuto messaggio SMS.

Fault (BOOL) Attivo per un loop se errore.

Text (STRING[160]) Testo del messaggio SMS ricevuto.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10003010 ***ModemID*** non definito.

10003020 ***ModemID*** non corretto.

Esempi

Nell'esempio è gestita la ricezione di un messaggio SMS dal modem definito nella variabile ***ModemID***, per la definizione delle variabili e per una migliore comprensione del funzionamento si rimanda agli esempi successivi.

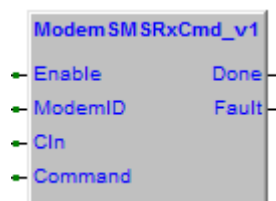
Esempio LD *(Ptp118a200, ReceiveSMSMessage)*

7.15.3 ModemSMSRxCmd_v1, receive a SMS command

Type	Library
FB	eModemLib_E000

Questo blocco funzione esegue la ricezione di un comando tramite un messaggio SMS, si collega al blocco funzione di gestione modem ***ModemCore***, occorre passare il ***ModemID*** in uscita dal blocco funzione di gestione modem.

Alla ricezione di un messaggio SMS se nel testo del messaggio è presente la stringa definita in ***Command***, si attiva per un loop di programma l'uscita ***Done***, all'uscita ***CLIPNumber*** della FB ***ModemCore*** è ritornato il numero di telefono da cui il messaggio è stato ricevuto.

Attivando ***Cin*** il controllo sulla stringa definita in ***Command*** verrà fatto non considerando il case (Maiuscolo/minuscolo) dei caratteri.

Enable (BOOL) Abilita la ricezione del comando.

ModemID (UDINT) ID modem fornito in uscita dalla ***ModemCore***.

CIn (BOOL) Se attivo, controllo di ***Command*** non considerando case (Maiuscolo/minuscolo) dei caratteri.

Command (@USINT) Pointer al testo del comando da eseguire.

Done (BOOL) Attivo per un loop se ricevuto messaggio SMS contenente il testo indicato in ***Command***.

Fault (BOOL) Attivo per un loop se errore.

Codici di errore

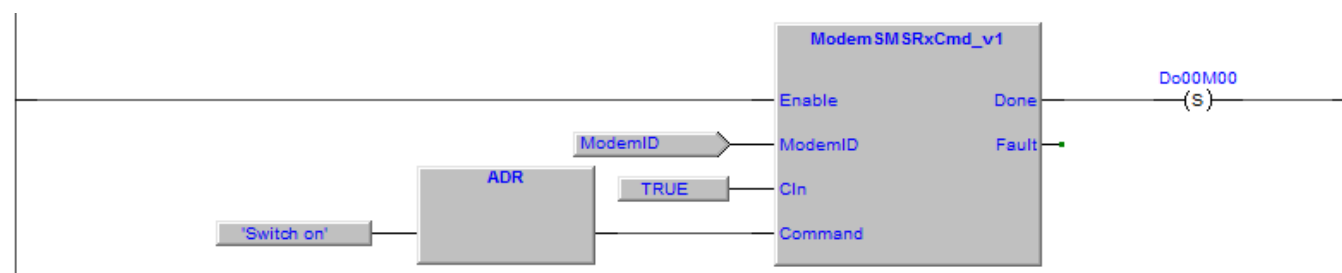
In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10004010 ***ModemID*** non definito.

10004020 ***ModemID*** non corretto.

Esempi

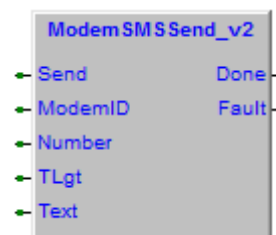
Nell'esempio è gestita la ricezione di un messaggio SMS dal modem definito nella variabile ***ModemID***, per la definizione delle variabili e per una migliore comprensione del funzionamento si rimanda agli esempi successivi.

Esempio LD *(Ptp118b100, ReceiveSMSCommand)*

7.15.4 ModemSMSSend_v2, send a SMS message

Type	Library
FB	eModemLib_E000

Questo blocco funzione esegue l'invio di un messaggio SMS, si collega al blocco funzione di gestione modem ***ModemCore***, occorre passare alla FB il ***ModemID*** in uscita dal blocco funzione di gestione modem.

Su fronte attivazione ingresso di ***Send*** viene prenotato l'invio del messaggio, non appena sarà possibile il messaggio definito in ***Text*** verrà inviato al numero definito in ***Number***. Terminato l'invio verrà attivata per un loop di programma l'uscita ***Done***.

TLgt definisce il numero di caratteri da inviare del messaggio definito in ***Text***, se non definito o definito “0” viene inviato l'intero messaggio ***Text***.

Send (BOOL) Sul fronte di attivazione comanda l'invio del messaggio SMS. ***Attenzione!*** Il messaggio sarà inviato non appena il modem è libero per l'invio.

ModemID (UDINT) ID modem fornito in uscita dalla ***ModemCore***.

Number (@USINT) Pointer al numero di telefono a cui eseguire l'invio del messaggio.

TLgt (UDINT) Numero di caratteri del messaggio da inviare. Se “0” viene inviato l'intero messaggio.

Text (@USINT) Pointer al testo messaggio da inviare.

Done (BOOL) Attivo per un loop al termine dell'invio del messaggio SMS.

Fault (BOOL) Attivo per un loop se errore invio messaggio SMS.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10005010 ***ModemID*** non definito.

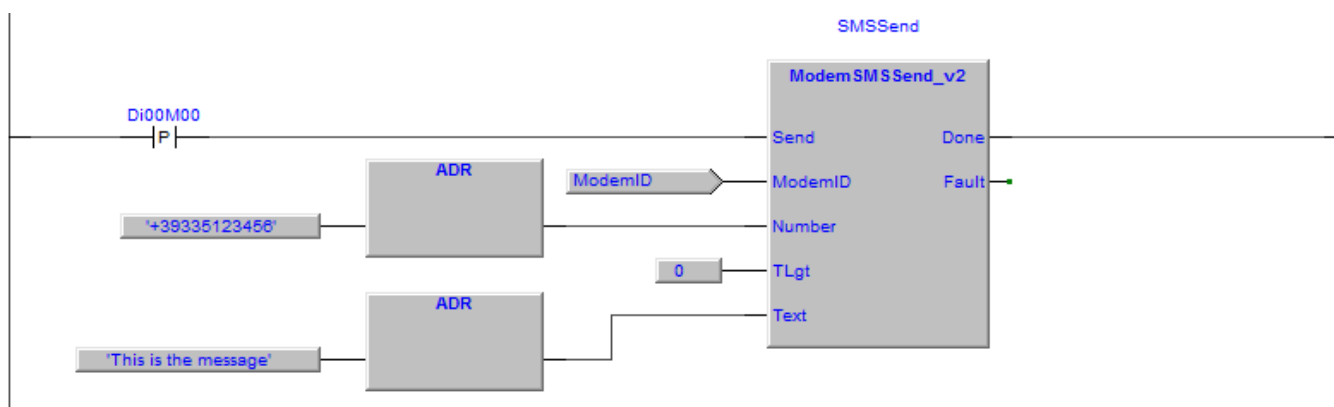
10005020 ***ModemID*** non corretto.

10005080 Errore case gestione.

10005100~3 Errore gestione invio SMS.

Esempi

Nell'esempio è gestito l'invio di un messaggio SMS sul modem definito nella variabile ***ModemID***, per la definizione delle variabili e per una migliore comprensione del funzionamento si rimanda agli esempi successivi.

Esempio LD *(Ptp118b100, SendSMSMessage)*

Invio di un messaggio a più numeri

Molte volte succede di dover inviare su un evento (Esempio ingresso digitale) diversi messaggi SMS ognuno con il proprio testo a diversi numeri telefonici. Il blocco funzione ***SMSSend*** permette la gestione dell'invio sullo stesso evento di più messaggi, ogni messaggio è contraddistinto dal numero di telefono e dal testo del messaggio.

Nell'esempio è gestito l'invio di un messaggio a più numeri di telefono. Come si può vedere attivando l'ingresso digitale ***Di00M00*** verranno inviati due messaggi a due diversi numeri di telefono. E' possibile aggiungere altri rami con il blocco funzione ***SMSSend*** ognuno con il proprio messaggio e numero di telefono sempre attivati da ***Di00M00***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Rssi	USINT	Auto	No		..	RSSI value
2	ModemID	UDINT	Auto	No		..	Modem ID
3	CLIP	STRING	Auto	[16]		..	CLIP received number
4	Operator	STRING	Auto	[16]		..	Operator value
5	Fp	FILEP	Auto	No		..	Serial pointer
6	Modem	ModemCore_v3	Auto	No		..	Modem core FB
7	SMSTest1	ModemSMSSend_v2	Auto	No		..	Modem SMS send FB
8	SMSTest2	ModemSMSSend_v2	Auto	No		..	Modem SMS send FB
9	SMode	SetSMode	Auto	No		..	Serial mode FB

Esempio LD *(Ptp118b100, MultipleSMSSend)*

SysFirstLoop
Sysfopen
EN ENO
'COM0' FName Fp
'rw' Mode

SMode
SysFirstLoop
SetSMode
EN ENO
Fp Fp Done
9600 Baudrate Fault
'N' Parity
8 DataBits
1 StopBits
DTR_ON DTRManagement
FALSE DTRComplement
FALSE EchoFlush
0 DTROnTime
0 DTROffTime

Modem

ModemCore_v3	
Enable	Enabled
AutoHangUp ← TRUE	Done
SpyOn ← TRUE	Fault
File ← Fp	PowerOn
APN	CLIPRxd
DNS	ModemID → ModemID
	Operator → Operator
	Rssi → Rssi
	CLIPNumber → CLIP
	IPAddress

SMSTest1

ModemSMSSend_v2	
Send ← Di00M00 (P)	Done
ModemID ← ModemID	Fault
Number ← ADR('+39335123456')	
TLgt ← 0	
Text ← ADR('This is the message')	

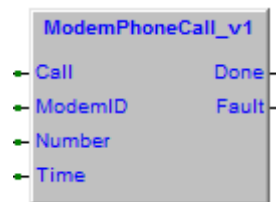
SMSTest2

ModemSMSSend_v2	
Send ← Di00M00 (P)	Done
ModemID ← ModemID	Fault
Number ← ADR('+39335654321')	
TLgt ← 0	
Text ← ADR('This is the message')	

7.15.5 ModemPhoneCall_v1, executes a phone call

Type	Library
FB	eModemLib_E000

Questo blocco funzione esegue una chiamata telefonica al numero indicato, si collega al blocco funzione di gestione modem ***ModemCore***, occorre passare il ***ModemID*** in uscita dal blocco funzione di gestione modem.

Su fronte attivazione ingresso di ***Call*** viene prenotata l'esecuzione della chiamata, non appena sarà possibile verrà eseguita la chiamata al numero definito in ***Number***. Terminato il tempo definito in ***Time*** la chiamata verrà terminata. Se non vi sono problemi verrà attivata per un loop di programma l'uscita ***Done***.

Call (BOOL) Sul fronte di attivazione comanda 'esecuzione della chiamata. ***Attenzione!*** La chiamata verrà eseguita non appena il modem è libero.

ModemID (UDINT) ID modem fornito in uscita dalla ***ModemCore***.

Number (@USINT) Pointer al numero di telefono da chiamare.

Time (UINT) Tempo di durata chiamata (Sec).

Done (BOOL) Attivo per un loop al termine della esecuzione chiamata.

Fault (BOOL) Attivo per un loop se errore esecuzione chiamata.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10037010 ***ModemID*** non definito.

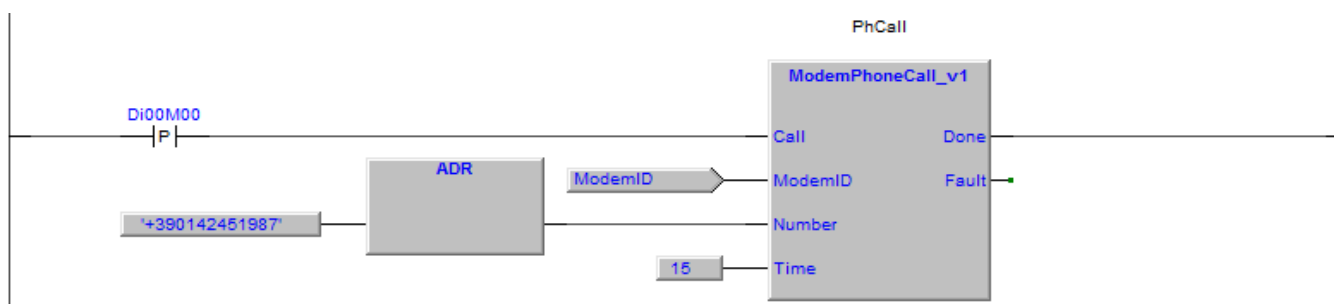
10037020 ***ModemID*** non corretto.

10037080 Errore case gestione.

10037100~3 Errore gestione chiamata telefonica.

Esempi

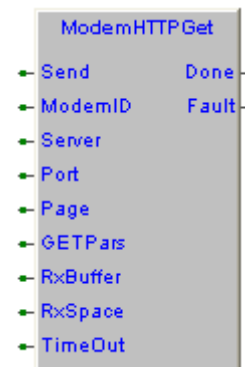
Nell'esempio è gestita una chiamata al numero indicato. Dopo 15 secondi la chiamata viene conclusa.

Esempio LD *(Ptp118b100, MakePhoneCall)*

7.15.6 ModemHTTPGet, executes a HTTP Get request

Type	Library
FB	eModemLib_E000

Questo blocco funzione esegue una richiesta HTTP inserendo in linea dei parametri GET, si collega al blocco funzione di gestione modem ***ModemCore***, occorre passare il ***ModemID*** in uscita dal blocco funzione di gestione modem.

Su fronte attivazione ingresso ***Send*** viene eseguita la connessione al ***Server*** HTPP sulla porta ***Port***, e richiesta la pagina ***Page***. La pagina viene richiesta con i parametri GET definiti nel buffer puntato da ***GETPars***. La pagina ricevuta dal server viene trasferita nel buffer puntato da ***RxBuffer***, i dati ricevuti che superano la dimensione del buffer ***RxSpace*** sono scartati.

Terminata la ricezione della pagina richiesta si attiva per un loop l'uscita ***Done***, se pagina non è ritornata nel tempo definito in ***TimeOut*** l'esecuzione termina con errore.

Send (BOOL) Sul fronte di attivazione comanda richiesta pagina HTTP. ***Attenzione!*** La richiesta verrà eseguita non appena il modem è libero.

ModemID (UDINT) ID modem fornito in uscita dalla ***ModemCore***.

Server (@USINT) Pointer alla stringa di definizione del server HTTP.

Port (UINT) Porta su cui effettuare la connessione.

Page (@USINT) Pointer alla stringa di definizione pagina richiesta.

GETPars (@USINT) Pointer alla stringa parametri GET da inserire nella richiesta (Max 512 caratteri).

RxBuffer (@USINT) Pointer al buffer di ricezione pagina richiesta (Max 360 caratteri).

RXSpace (UINT) Dimensione in bytes del buffer di ricezione pagina.

TimeOut (UINT) Tempo massimo attesa ricezione pagina (mS).

Done (BOOL) Attivo per un loop al termine della ricezione pagina.

Fault (BOOL) Attivo per un loop se errore ricezione pagina.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10042010 ***ModemID*** non definito.

10042020 ***ModemID*** non corretto.

10042080 Errore case gestione.

10042100~8 Errore gestione richiesta pagina da server HTTP.

10042200 Stringa GET parametri troppo lunga.

Esempi

L'utilizzo tipico di questa FB è di collegarsi ad un server HTTP dove sono eseguite pagine script (ASP, PHP, Phyton, ecc) richiedendo la pagina fornendo in linea alla chiamata dei parametri in GET. Lo script della pagina potrà eseguire operazioni con i dati passati in linea e ritornare dei valori come risultato. Ecco un semplice esempio che tramite una pagina script PHP sul server permette di inviare una Email.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Pulse	BOOL	Auto	No		..	Auxiliary pulse
2	Result	BOOL	Auto	No		..	Result flag
3	ABf	INT	Auto	No		..	Auxiliary buffer
4	RxD	STRING	Auto	[32]		..	HTTP Rx data
5	TxD	STRING	Auto	[128]		..	HTTP Tx data
6	Mdm	ModemCore_v3	Auto	No		..	Modem management
7	HTTP	ModemHTTPGet	Auto	No		..	HTTP Get
8	Sm	SYSSERIALMODE	Auto	No		..	Serial mode

Esempio ST *(Ptp118b100, EmailWithHTTPGet)*

```
(* ************************************************************************** *)
(* PROGRAM "EmailWithHTTPGet"                                                 *)
(* ************************************************************************** *)
(* This program sends an Email by contacting a PHP script.                    *)
(* -------------------------------------------------------------------------- *)

    (* ---------------------------------------------------------------------- *)
    (* INITIALIZATION                                                         *)
    (* ---------------------------------------------------------------------- *)
    (* General initializations. *)

    IF (SysFirstLoop) THEN

        (* Opening the serial port on wich the modem is connected. *)

        Mdm.File:=Sysfopen('COM0', 'rw'); (* COM port file pointer *)

        (* Set the serial communication mode. *)

        ABf:=SysGetSerialMode(ADR(Sm), Mdm.File);
        Sm.Baudrate:=115200; (* Baudrate *)
        Sm.Parity:='N'; (* Parity *)
        Sm.DataBits:=8; (* Data bits *)
        Sm.StopBits:=1; (* Stop bits *)
        Sm.DTRManagement:=DTR_OFF; (* DTR management *)
        Sm.DTRComplement:=FALSE; (* DTR complement *)
        Sm.EchoFlush:=TRUE; (* Flush the echo caracters *)
        ABf:=SysSetSerialMode(ADR(Sm), Mdm.File);

        (* Modem parameters. Please change them according your requirements. *)

        Mdm.AutoHangUp:=TRUE; (* Auto hangup *)
        Mdm.APN:=ADR('M2Mbis.vodafone.it'); (* APN definition *)
        Mdm.DNS:=ADR('8.8.8.8'); (* DNS definition *)

        (* HTTP parameters. Please change them according your requirements. *)

        HTTP.Server:=ADR('www.MyServer.com'); (* HTTP server *)
        HTTP.Port:=80; (* HTTP port *)
        HTTP.Page:=ADR('eMailSend.php'); (* Page name *)
        HTTP.TimeOut:=5000; (* Connection timeout (mS) *)
        HTTP.GETPars:=ADR(TxD); (* GET parameters pointer *)
        HTTP.RxBuffer:=ADR(RxD); (* Rx buffer pointer *)
        HTTP.RxSpace:=SIZEOF(RxD); (* Rx buffer space *)
    END_IF;

    (* ---------------------------------------------------------------------- *)
    (* FBs MANAGEMENT                                                         *)
    (* ---------------------------------------------------------------------- *)
    (* Manage the modem core. *)
```

```
    Mdm(Enable:=TRUE); (* Modem core management *)
    HTTP.ModemID:=Mdm.ModemID; (* Modem ID *)

    (* Manage the HTTP connection. *)

    HTTP();
    HTTP.Send:=FALSE;

    (* -------------------------------------------------------------------- *)
    (* GET THE HTTP PAGE AND SEND THE EMAIL                                 *)
    (* -------------------------------------------------------------------- *)
    (* On digital input activation the HTTP page is requested. *)

    IF (Di00M00 <> Pulse) THEN
        Pulse:=Di00M00; (* Auxiliary pulse *)

        IF (Pulse) THEN
            ABf:=SysVarsnprintf(ADR(TxD), 128, 'To=%s', STRING_TYPE, ADR('sbertana@elsist.it'));
            ABf:=SysVarsnprintf(ADR(TxD)+LEN(TxD), 128-LEN(TxD), '$26Subject=%s', STRING_TYPE,
ADR('Subject'));
            ABf:=SysVarsnprintf(ADR(TxD)+LEN(TxD), 128-LEN(TxD), '$26Text=%s', STRING_TYPE, ADR('Text'));
             HTTP.Send:=TRUE;
        END_IF;
    END_IF;

    (* Check the HTTP request answer. *)

    IF (HTTP.Done) THEN
        IF (FIND(RxD, 'Ok') = 0) THEN Result:=FALSE; ELSE Result:=TRUE; END_IF;
    END_IF;

(* [End of file] *)
```

Vediamo di spiegare come funziona il programma, sul fronte di attivazione dell'ingresso ***Di00M00*** viene effettuata la connessione al server ***www.MyServer.com*** e richiesta l pagina ***eMailSend.php***. In linea alla chiamata della pagina sono posti i campi GET **To=sbertana@elsist.it&Subject=Subject&Text=Text**. Da notare che il carattere “&” è stato sostituito da “$26”.

La pagina ***eMailSend.php*** sul server contiene uno script PHP del tipo:

```
<?php
    $Headers="MIME-Version: 1.0\n";
    $Headers.="Content-type: text/html; charset=iso-8859-1\n";
    $Headers.="To: ".$_REQUEST['To']."\n";
    $Headers.="From: \n";
    mail($_REQUEST['To'], $_REQUEST['Subject'], $_REQUEST['Text'], $Headers);
    echo "Ok";
    exit();
?>
```

Come si vede lo script invia una mail tramite il comando mail i cui campi sono valorizzati dai parametri posti in GET alla richiesta. Lo statement ***$_REQUEST['To']*** è valorizzato con il testo definito in ***To=***, ***$_REQUEST['Subject']*** è valorizzato con il testo definito in ***Subject=***, e così via. Come si vede è possibille passare allo script tutti i valori che si desiderano.

Il valore di ritorno dallo script definito con lo statement echo verrà trasferito nel buffer ***RxD*** del nostro programma e quindi è possibile operare su di esso con le istruzioni di gestione stringa.

7.16 Funzioni ed FB gestione One-Wire (ePLC1WireLib_C000)

Attenzione! Per utilizzare la libreria occorre importarla nel proprio progetto. Vedere capitolo relativo all'import delle librerie.

La rete da campo 1 Wire® é un protocollo standard basato su di un solo filo di comunicazione, come indica lo stesso nome, che include numerosi dispositivi e sensori frequentemente utilizzati nel campo dell'automazione industriale e domestica.

I dispositivi sono interconnessi da soli due fili, uno per la massa ed uno per il segnale e l'alimentazione; su questi due fili possono essere collegati tutti i dispositivi in rete scegliendo la disposizione fisica necessaria.

Il protocollo 1 Wire® é dotato di tutte le modalità di comunicazione che consentono di ottenere un elevato trasferimento dati ed una sicurezza intrinseca sulla loro validità. Questo avviene grazie a tecniche di indirizzamento univoche, CRC polinomiali di controllo, numerosi comandi di verifica e complessi algoritmi di gestione.

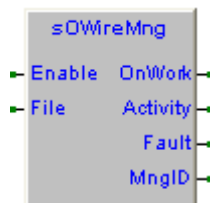
Sui sistemi SlimLine è possibile connettere un convertitore Seriale/One-Wire su una delle porte seriali COM0 o COM1 presenti sul modulo CPU, oppure utilizzare una qualunque porta seriale presente su un modulo di estensione per poter disporre di uno o più bus One-Wire.

Il blocco funzione ***sOWireMng*** gestisce il convertitore permettendo la gestione dei dispositivi ad esso connessi.

7.16.1 sOWireMng, One-Wire management

Type	Library
FB	ePLC1WireLib_C000

Questo blocco funzione gestisce il convertitore Seriale/One-Wire connesso al dispositivo di I/O definito in ***File***, l'FB gestisce l'inizializzazione e la gestione del convertitore.

L'uscita ***OnWork*** si attiva se ***Enable*** attivo, l'uscita ***Fault*** si attiva in caso di errori di gestione e rimane attiva fino a quando l'errore permane. L'uscita ***Activity*** si attiva durante l'attività sul bus One-Wire. L'FB ritorna un ***MngID*** che deve essere passato alle FB collegate.

Enable (BOOL) Abilitazione blocco funzione, attivandolo viene gestito il convertitore Seriale/One-Wire.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

OnWork (BOOL) Function block attivata ed in lavoro.

Activity (BOOL) Attivo se attività sul bus One-Wire.

Fault (BOOL) Attivo se errore gestione bus One-Wire.

MngID (UDINT) One-Wire management ID, da passare alle FB collegate (Esempio ***sOWRdIdentifier***, ***sOWRdTemperature***, ecc.).

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10008010 Valore di ***File*** non definito.

10008200 Timeout invio stringa comando.

10008300~9 Errore nelle configurazione convertitore.

10008400~1 Errore nella verifica configurazione.

10008500~2 Errore nelle sequenze indirizzamento device One-Wire.

10008600 Errore risposta a comando di reset pulse.

10008601 Bus One-Wire in cortocircuito.

10008602 Errore dispositivi su bus One-Wire.

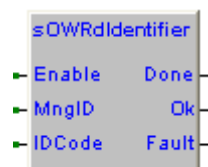
10008603 Nessun dispositivo su bus One-Wire.

7.16.2 sOWRdIdentifier, One-Wire read ROM identifier

Type	Library
FB	ePLC1WireLib_C000

Questo blocco funzione esegue la lettura del codice di identificazione di un dispositivo One-Wire, si collega al blocco funzione ***sOWireMng*** di gestione convertitore Seriale/One-Wire. Occorre passare ***MngID*** in uscita dal blocco funzione di gestione convertitore.

Attivando ***Enable***, viene eseguita la lettura del ROM ID dal dispositivo connesso al bus One-Wire ***Attenzione! Bisogna avere un solo dispositivo connesso al bus***. Al termine della lettura del codice si attiva l'uscita ***Done***. Se la lettura ha esito positivo si attiva per un loop di programma l'uscita ***Ok***. Gli 8 bytes del ROM ID sono trasferiti nell'array indirizzato da ***IDCode***. Disattivando ***Enable*** si azzera ***Done*** e l'eventuale ***Fault***, per eseguire una nuova lettura occorre riabilitare l'ingresso ***Enable***.

sOWRdIdentifier: Enable, MngID, IDCode → Done, Ok, Fault

Enable (BOOL) Abilita il blocco funzione.

MngID (UDINT) One-Wire management ID, fornito in uscita dal blocco funzione ***sOWireMng***.

IDCode (@USINT) Puntatore array memorizzazione ROM ID, deve essere almeno 8 bytes.

Done (BOOL) Si attiva al termine della esecuzione lettura del ROM ID.

Ok (BOOL) Attivo per un loop se lettura ROM ID eseguita correttamente.

Fault (BOOL) Attivo se errore lettura ROM ID.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10009010 ***MngID*** non definito.

10009020 ***MngID*** non corretto.

10009030 ***IDCode*** non definito.

10009100 FB ***OWireMng***, gestione convertitore Seriale/One-Wire, impegnata.

10009200~2 Errore gestione sequenze One-Wire lettura ID.

Esempi

Ecco un semplice esempio di gestione di dispositivi iButton per il riconoscimento personale. Inserendo il TAG nel lettore viene eseguita la lettura del ROM identifier, il valore acquisito è trasferito in un array di 8 bytes ***ROMID***.

Ciclicamente viene eseguita l'acquisizione, se un TAG è inserito nel lettore viene attivato ***TAGInserted***.

Per semplicità nel programma non viene eseguito alcun controllo sull'ID letto, ma in un sistema di controllo accessi ad esempio è possibile dall'ID letto identificare la persona ed abilitare o no l'accesso.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	IDDone	BOOL	Auto	No	FALSE	..	Read ID code done
2	IDOk	BOOL	Auto	No	FALSE	..	ID code read
3	TAGInserted	BOOL	Auto	No	FALSE	..	TAG inserted on reader
4	ROMID	BYTE	Auto	[0..7]	8(0)	..	ROM ID code
5	OwID	UDINT	Auto	No	0	..	One-Wire ID
6	Fp	FILEP	Auto	No	0	..	File pointer
7	OWire	sOWireMng	Auto	No	0	..	FB One Wire management
8	RdID	sOWRdIdentifier	Auto	No	0	..	FB One Wire read ID

Esempio LD *(PTP120A200, LD_TAGReader)*

Open the serial port

SysFirstLoop — Sysfopen (EN, ENO; FName: 'COM0'; Mode: 'rw') → Fp

Manage the Serial/One-Wire converter

OWire: sOWireMng — SysFirstLoop (NC) → Enable; Fp → File; OnWork, Activity, Fault; MngID → OwID

Execute the TAG reader code

RdID: sOWRdIdentifier — IDDone (NC) → Enable; OwID → MngID; ROMID → ADR → IDCode; Done → (IDDone); Ok → (IDOk); Fault

IDDone, IDOk (NC) → (R) TAGInserted

IDDone, IDOk → (S) TAGInserted

7.16.3 sOWRdTemperature, One-Wire read temperature

Type	Library
FB	ePLC1WireLib_C000

Questo blocco funzione esegue l'acquisizione di un sensore One-Wire di temperatura (Maxim DS18B20), si collega al blocco funzione ***sOWireMng*** di gestione convertitore Seriale/One-Wire. Occorre passare ***MngID*** in uscita dal blocco funzione di gestione convertitore.

Attivando ***Enable***, viene eseguita la lettura del valore di temperatura dal dispositivo connesso al bus One-Wire, terminata l'acquisizione si attiva l'uscita ***Done***, se acquisizione corretta si attiva per un loop l'uscita ***Ok*** e su ***Temperature***, sarà riportato il valore di temperatura acquisito.

L'uscita ***Fault*** si attiva in caso di errori di gestione. Disattivando ***Enable*** si azzera ***Done*** e l'eventuale ***Fault***, per eseguire una nuova lettura occorre riabilitare l'ingresso ***Enable***.

Se sul bus One-Wire è connesso un unico dispositivo, si può evitare di settare ***IDCode*** oppure si può forzare a ***0***. Se invece sul bus One-Wire sono presenti più dispositivi parallelati, in ***IDCode*** occorre definire l'indirizzo dell'array di 8 bytes che contiene il ROM ID del dispositivo che si vuole acquisire.

Enable (BOOL) — Abilita il blocco funzione.

MngID (UDINT) — One-Wire management ID, fornito in uscita dal blocco funzione ***sOWireMng***.

IDCode (@USINT) — Puntatore ad array definizione ROM ID dispositivo da acquisire. Se 0 viene eseguita lettura con comando Skip ROM (Viene acquisito qualsiasi device connesso).

Done (BOOL) — Si attiva al termine della esecuzione lettura temperatura.

Ok (BOOL) — Attivo per un loop se lettura temperatura eseguita correttamente.

Fault (BOOL) — Attivo se errore lettura temperatura.

Temperature (REAL) — Valore di temperatura acquisito (°C). Range di lettura da -55 (°C) a +125 (°C). Precisione ±0.5 (°C) tra -10 (°C) e +85 (°C). Risoluzione di lettura 0.0625 (°C).

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10010010 ***MngID*** non definito.

10010020 ***MngID*** non corretto.

10010100 FB ***OWireMng***, gestione convertitore Seriale/One-Wire, impegnata.

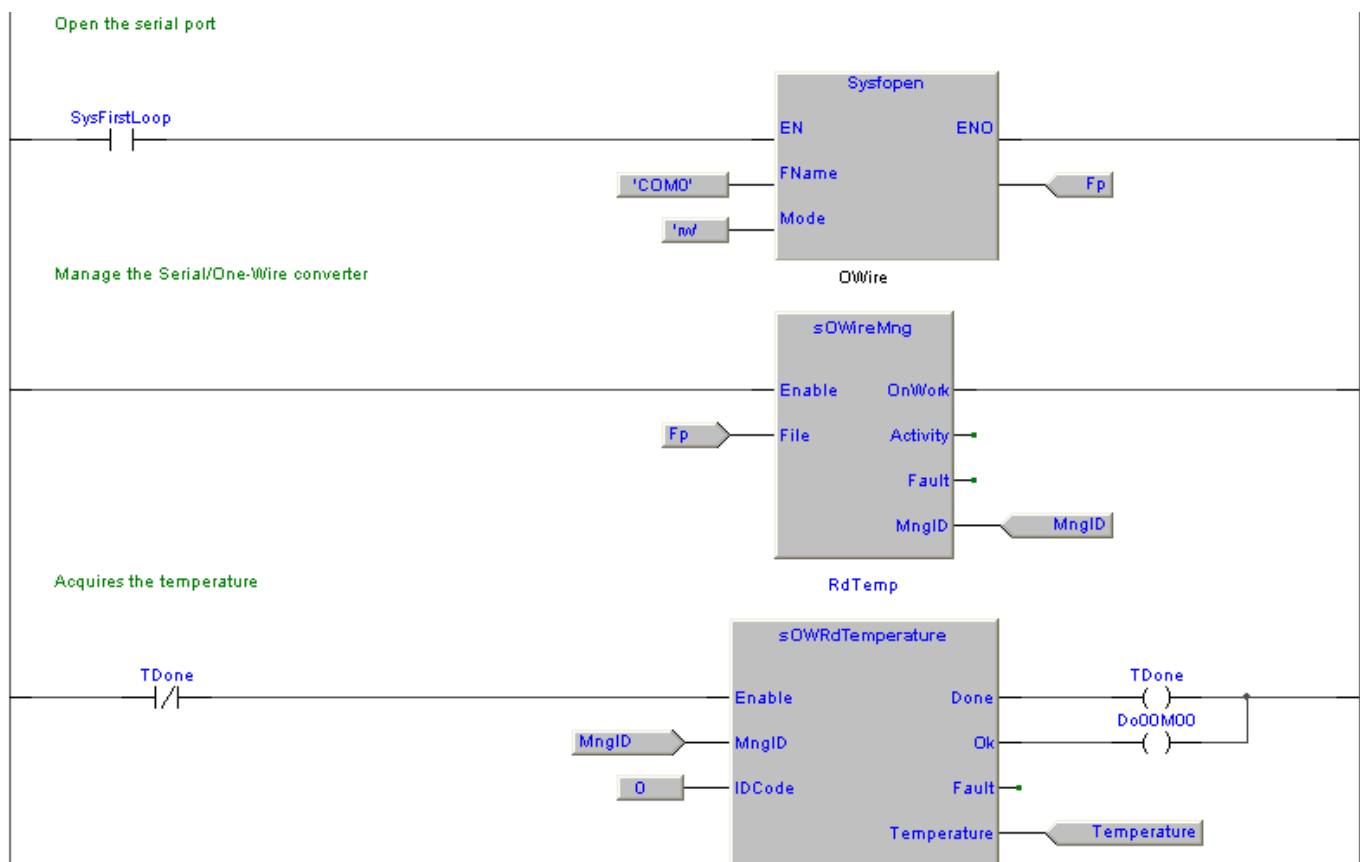
10010200~5 Errore nelle sequenze acquisizione temperatura.

Esempi

Viene eseguita la lettura della temperatura da un dispositivo One-Wire. Non essendo definito ***IDCode*** viene acquisito qualsiasi dispositivo presente sul bus One-Wire ***Attenzione! Deve essere presente un solo dispositivo sul bus***. Se esecuzione corretta viene attivata l'uscita ***Do00M00*** ed il valore acquisito è trasferito nella variabile ***Temperature***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	MngID	UDINT	Auto	No	0	..	One-Wire management ID
3	Temperature	REAL	Auto	No	0	..	Temperature (°C)
4	RdTemp	sOWRdTemperature	Auto	No	0	..	Read temperature FB
5	TDone	BOOL	Auto	No	FALSE	..	Read temperature done
6	OWire	sOWireMng	Auto	No	0	..	One Wire management FB

Esempio LD *(PTP120A200, LD_SingleTempSkipROM)*

7.16.4 sOWRdHumidity, One-Wire read humidity

Type	Library
FB	ePLC1WireLib_C000

Questo blocco funzione esegue gestione di un sensore One-Wire di acquisizione umidità e temperatura basato (Maxim DS2438). Si collega al blocco funzione ***OWireMng*** di gestione convertitore Seriale/One-Wire. Occorre passare ***MngID*** in uscita dal blocco funzione di gestione convertitore.

Attivando ***Enable***, viene eseguita la lettura del valore di umidità e temperatura dal dispositivo connesso al bus One-Wire, terminata l'acquisizione si attiva l'uscita ***Done***, se acquisizione corretta si attiva per un loop l'uscita ***Ok*** e su ***Humidity*** e ***Temperature***, sarà riportato il valore di temperatura acquisito.

L'uscita ***Fault*** si attiva in caso di errori di gestione. Disattivando ***Enable*** si azzera ***Done*** e l'eventuale ***Fault***, per eseguire una nuova lettura occorre riabilitare l'ingresso ***Enable***.

Se sul bus One-Wire è connesso un unico dispositivo, si può evitare di settare ***IDCode*** oppure si può forzare a ***0***. Se invece sul bus One-Wire sono presenti più dispositivi parallelati, in ***IDCode*** occorre definire l'indirizzo dell'array di 8 bytes che contiene il ROM ID del dispositivo che si vuole acquisire.

Enable (BOOL) Abilita il blocco funzione.

MngID (UDINT) One-Wire management ID, fornito in uscita dal blocco funzione ***sOWireMng***.

IDCode (@USINT) Puntatore ad array definizione ROM ID dispositivo da acquisire.

Done (BOOL) Si attiva al termine della esecuzione lettura umidità e temperatura.

Ok (BOOL) Attivo per un loop se lettura umidità e temperatura eseguita correttamente.

Fault (BOOL) Attivo se errore lettura umidità e temperatura.

Temperature (REAL) Valore di temperatura acquisito (°C). Range di lettura da -55 (°C) a +125 (°C). Precisione ±0.5 (°C) tra -10 (°C) e +85 (°C). Risoluzione di lettura 0.03125 (°C).

Humidity (REAL) Valore di umidità acquisito (RH%).

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10015010 ***MngID*** non definito.

10015020 ***MngID*** non corretto.

10015100 FB ***OWireMng***, gestione convertitore Seriale/One-Wire, impegnata.

10015200~3 Errore nelle sequenze conversione temperatura.

10015300~8 Errore nelle sequenze acquisizione tensione alimentazione sensore.

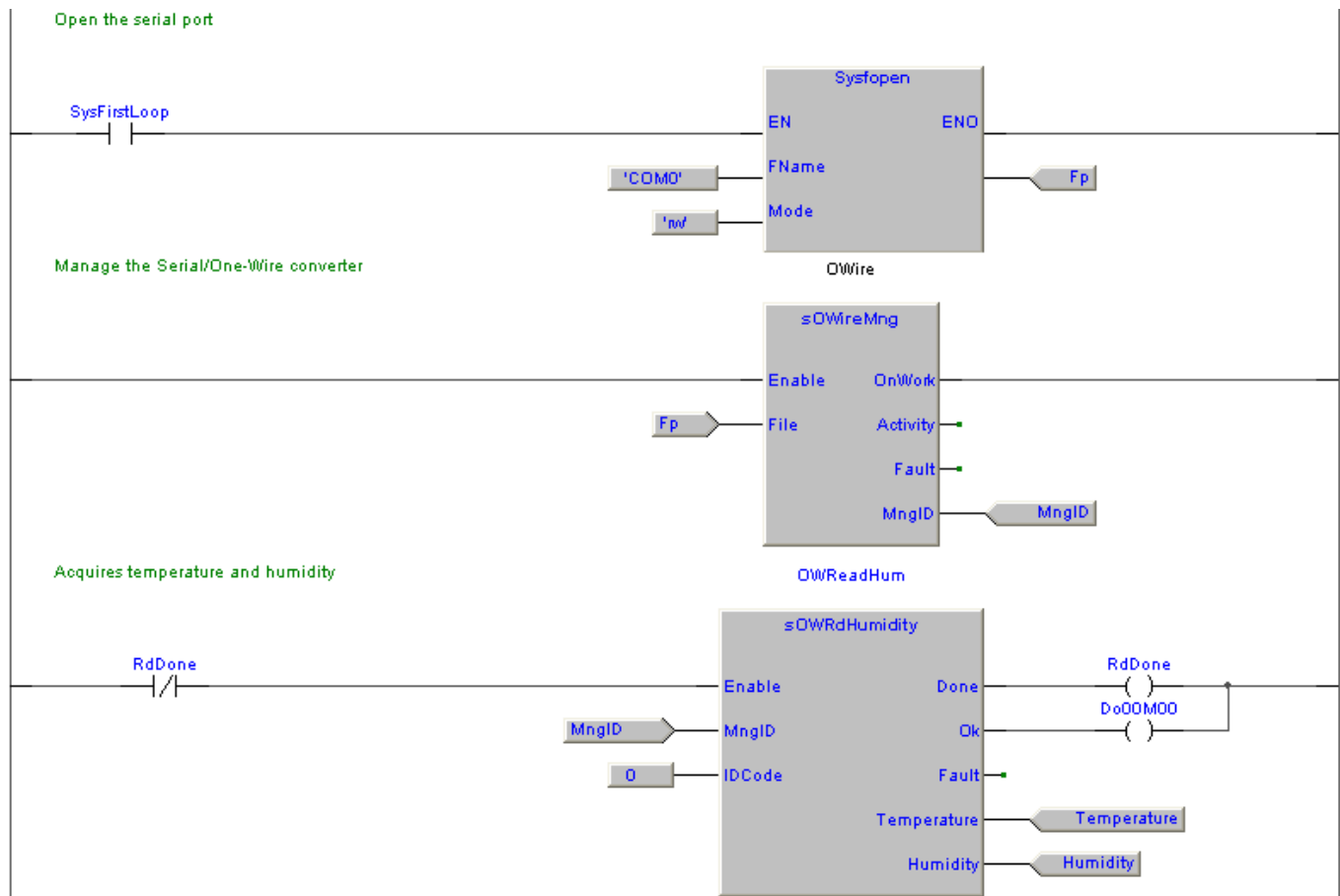
10015400~8 Errore nelle sequenze acquisizione sensore umidità.

Esempi

Viene eseguita la lettura della temperatura ed umidità da un dispositivo One-Wire. Non essendo definito ***IDCode*** viene acquisito qualsiasi dispositivo presente sul bus One-Wire ***Attenzione! Deve essere presente un solo dispositivo sul bus***. Se acquisizione corretta viene attivata l'uscita ***Do00M00*** ed i valori acquisiti sono trasferiti nelle variabili ***Temperature*** e ***Humidity***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RdDone	BOOL	Auto	No	FALSE	..	Read humidity/temperature done
2	MngID	UDINT	Auto	No	0	..	One-Wire management ID
3	Humidity	REAL	Auto	No	0	..	Humidity (RH%)
4	Temperature	REAL	Auto	No	0	..	Temperature (°C)
5	Fp	FILEP	Auto	No	0	..	File pointer
6	OWire	sOWireMng	Auto	No	0	..	One-Wire management FB
7	OWReadHum	sOWRdHumic	Auto	No	0	..	FB read humidity data

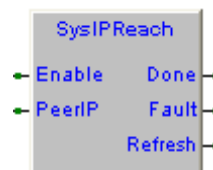
Esempio LD *(PTP120A200, LD_HumiditySkipROM)*

7.17 Funzioni ed FB gestione networking

Da linguaggio IEC sono disponibili funzioni e blocchi funzione per la gestione networking.

7.17.1 SysIPReach, IP address is reachable

Type	Library
FB	XTarget_07_0

Questo blocco funzione esegue controllo se un indirizzo IP è raggiungibile, viene eseguito l'invio del comando Ping al sistema e se si ottiene risposta viene attivato ***Done***.

La variabile ***Refresh*** ritorna la percentuale di tempo trascorsa dall'ultima esecuzione del comando Ping sul sistema. Raggiunto il 50% del tempo circa 25 Secondi viene eseguito un nuovo comando Ping.

Enable (BOOL) Abilitazione blocco funzione, attivandolo viene eseguito un ping ogni 25 Sec

PeerIP (STRING[15]) Stringa di definizione indirizzo IP di cui eseguire la ricerca.

Done (BOOL) Attivo se indirizzo IP è raggiungibile (Risposta da Ping).

Fault (BOOL) Attivo per un loop di programma se errore gestione.

Refresh (USINT) Percentuale di tempo da ultima esecuzione Ping.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

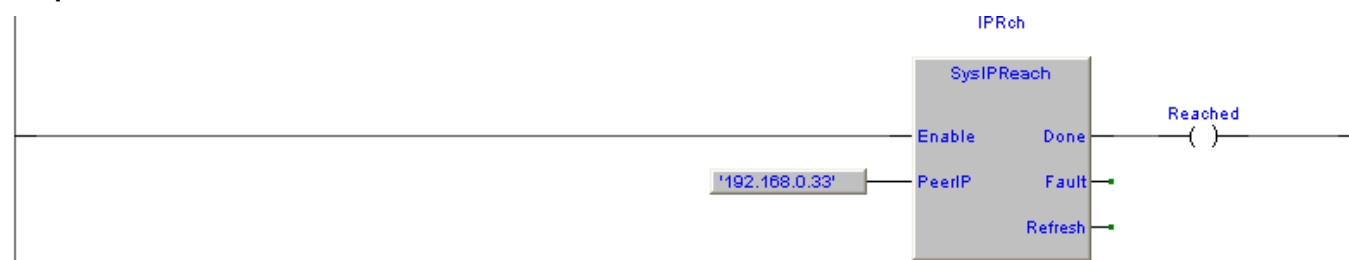
Codice	Descrizione
9974005	Blocco funzione non supportato.
9974050	Errore allocazione blocco funzione.
9974060	Terminato spazio memoria rilocabile, non è possibile eseguire l''FB.
9974070	Errore versione blocco funzione.
9974100	Blocco funzione eseguito in una task diversa da Back.
9974110	Errore indirizzo IP definito ***PeerIP***.
9974200	Errore invio richiesta Ping.
9974300	Errore risposta a richiesta Ping.
9974990	Non implementata nel simulatore.

Esempi

Nell'esempio viene controllato se l'indirizzo IP ***192.168.0.33*** è raggiungibile, in tal caso si attiva l'uscita ***Reached***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Reached	BOOL	Auto	No	FALSE	..	IP address reached
2	IPRch	SysIPReach	Auto	No	0	..	FB IP reach

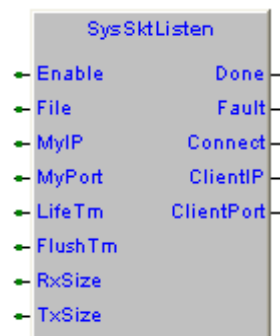
Esempio LD

7.17.2 SysSktListen, Socket listen

Type	Library
FB	XTarget_07_0

Questo blocco funzione forza il socket nella condizione di listening, occorre passare alla FB un flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

I parametri ***MyIP*** e ***MyPort*** indicano l'interfaccia di rete e la porta su cui il socket è in ascolto, ***MyIP*** può essere lasciato non connesso ed in tal caso viene utilizzata l'interfaccia ethernet. I parametri ***LifeTm***, ***FlushTm***, ***RxSize***, ***TxSize***, devono essere definiti solo se si tratta di socket TCP, possono essere lasciati non connessi nel caso di socket UDP.

L'uscita ***Done*** si attiva se tutti i parametri sono corretti ed il socket è stato messo in condizione di listening. Quando un client si connette (Solo socket TCP) si attiva l'uscita ***Connect*** e nelle variabili ***ClientIP*** e ***ClientPort*** sono ritornati l'indirizzo IP e la porta connessa al socket.

SysSktListen
Enable — Done
File — Fault
MyIP — Connect
MyPort — ClientIP
LifeTm — ClientPort
FlushTm
RxSize
TxSize

Enable (BOOL) Abilitazione blocco funzione, attivandolo si forza socket in condizione di listening.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

MyIP (STRING[15]) Stringa di definizione indirizzo IP interfaccia di rete su cui il socket è gestito. Lasciandolo non connesso o definendo ***'255.255.255.255'*** sarà utilizzata interfaccia ethernet.

MyPort (UINT]) Porta su cui sarà posto in ascolto il socket.

LifeTm (UINT]) Tempo di vita socket, se non sono ricevuti o inviati dati dopo il tempo definito il socket viene automaticamente chiuso (Sec) (Solo per socket TCP).

FlushTm (UINT]) Tempo di flush dati, se non sono caricati dati nel socket dopo il tempo definito i dati presenti vengono automaticmente inviati (mS) (Solo per socket TCP).

RxSize (UINT]) Dimensione buffer ricezione dati (Solo per socket TCP).

TxSize (UINT]) Dimensione buffer trasmissione dati (Solo per socket TCP).

Done (BOOL) Attivo se socket in condizione di listening.

Fault (BOOL) Attivo per un loop di programma se errore gestione.

Connect (BOOL) Si attiva se un client si connette al socket (Solo per socket TCP).

ClientIP (STRING[15]) Stringa di definizione indirizzo IP del client connesso al socket (Solo per socket TCP).

ClientPort (UINT]) Porta del client attualmente connesso al socket (Solo per socket TCP).

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

Codice	Descrizione
9977005	Blocco funzione non supportato.
9977050	Errore allocazione blocco funzione
9977060	Terminato spazio memoria rilocabile, non è possibile eseguire l'"FB.
9977070	Errore versione blocco funzione.
9977100	Valore di ***File*** non definito.
9977110	Tipo di ***stream*** definito in ***File*** non corretto.
9977150	Errore indirizzo IP dispositivo client ***ClientIP***.
9977200	Errore indirizzo IP interfaccia di rete ***MyIP***.
9977300	Valore dimensione buffer ricezione ***RxSize*** fuori range.
9977310	Valore dimensione buffer trasmissione ***TxSize*** fuori range.
9977350	Errore nel set dei parametri TCP.
9977400	Errore set socket in listening.
9977990	Non implementata nel simulatore.

Esempi

Nell'esempio è posto in ascolto un socket TCP sulla porta 1000. Se ci si connette con un client telnet (Esempio Hyperterminal) al sistema SlimLine sulla porta 1000, si attiverà la variabile ***SktConnect***. Inviando caratteri dalla finestra del terminale telnet, si attiverà la variabile ***ChrsAv***, i caratteri inviati saranno ricevuti dalla funzione ***Sysfgetc*** e inviati in echo verso il client telnet dalla funzione ***Sysfputc***.

Definizione variabili

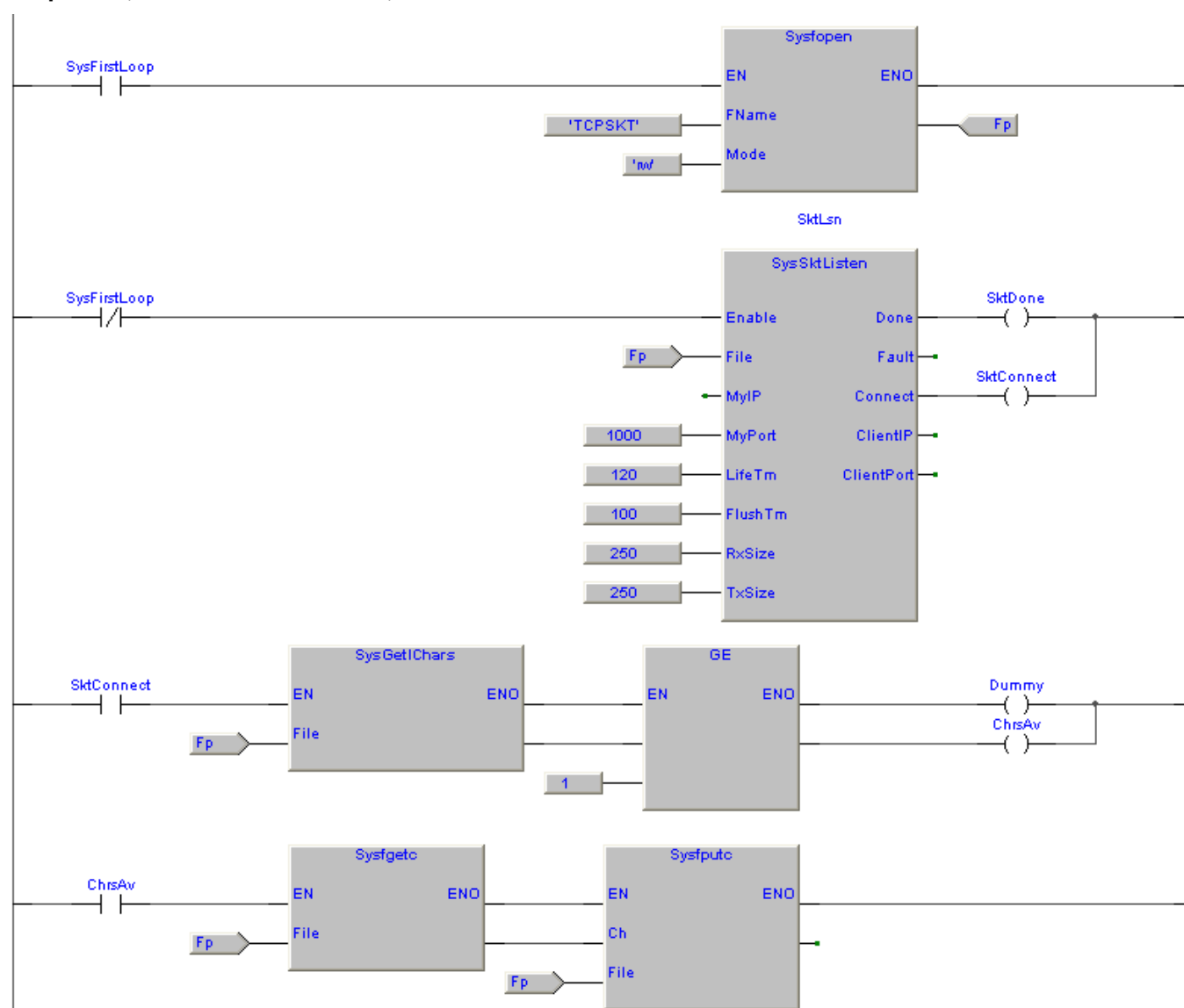
	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	SktLsn	SysSktListen	Auto	No	0	..	Socket listen FB
3	SktDone	BOOL	Auto	No	FALSE	..	Socket done
4	SktConnect	BOOL	Auto	No	FALSE	..	Socket connect
5	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
6	ChrsAv	BOOL	Auto	No	FALSE	..	Characters available

Esempio LD *(PTP119A200, TCPSocketEcho)*

SysFirstLoop
Sysfopen
EN ENO
'TCPSKT' FName
'rw' Mode
Fp

SktLsn
SysSktListen
SysFirstLoop
Enable Done SktDone
Fp File Fault
MyIP Connect SktConnect
1000 MyPort ClientIP
120 LifeTm ClientPort
100 FlushTm
250 RxSize
250 TxSize

SktConnect
SysGetIChars
EN ENO
Fp File
GE
EN ENO
1
Dummy
ChrsAv

ChrsAv
Sysfgetc
EN ENO
Fp File
Sysfputc
EN ENO
Ch
Fp File

7.17.3 SysUDPSktRcv, UDP socket receive

Type	Library
FB	XTarget_07_0

Questo blocco funzione esegue la ricezione dati da un socket UDP, occorre passare alla FB un flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen*** ed il socket deve essere stato posto in condizione di listening dalla FB ***SysSktListen***.

SysUDPSktRcv
Enable Done
File Fault
PeerIP ClientIP
PeerPort ClientPort
RxData RxSize
Size

I parametri ***PeerIP*** e ***PeerPort*** indicano l'indirizzo IP e la porta da cui sono accettati i dati, se non connessi sono accettate connessioni da tutti gli IP e da tutte le porte.

Nel parametro ***RxData*** bisogna definire l'indirizzo del buffer di memoria in cui verranno trasferiti i dati ricevuti, ed in ***Size*** occorre definire la dimensione del buffer.

Su ricezione dati si attiva l'uscita ***Done*** e nelle variabili ***ClientIP*** e ***ClientPort*** sono ritornati l'indirizzo IP e la porta da cui sono stati ricevuti i dati, mentre in ***RxSize*** è ritornato il numero di bytes dati ricevuti.

Enable (BOOL) — Abilitazione blocco funzione.

File (FILEP) — Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

PeerIP (STRING[15]) — Stringa di definizione indirizzo IP da cui saranno accettate le connessioni. Lasciandolo non connesso o definendo ***'255.255.255.255'*** saranno accettate connessioni da tutti gli IP.

PeerPort (UINT]) — Porta da cui saranno accettate le connessioni, Lasciandolo non connesso o definendo ***65536*** saranno accettate connessioni da tutte le porte.

RxData (@USINT]) — Puntatore al buffer dove devono essere trasferiti i dati ricevuti.

Size (UINT]) — Dimensione buffer ricezione dati.

Done (BOOL) — Attivo per un loop di programma su ricezione dati.

Fault (BOOL) — Attivo per un loop di programma se errore gestione.

ClientIP (STRING[15]) — Stringa di definizione indirizzo IP del client da cui sono stati ricevuti i dati.

ClientPort (UINT]) — Porta del client da cui sono stati ricevuti i dati.

RxSize (UINT]) — Numero di bytes di dato ricevuti.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

Codice	Descrizione
9975005	Blocco funzione non supportato.
9975050	Errore allocazione blocco funzione
9975060	Terminato spazio memoria rilocabile, non è possibile eseguire l''FB.
9975070	Errore versione blocco funzione.
9975100	Valore di ***File*** non definito.
9975110	Tipo di ***stream*** definito in ***File*** non corretto.
9975200	Errore indirizzo IP connessioni accettate ***PeerIP***.
9975300	Errore ricezione dati da socket UDP
9975350	Errore indirizzo IP client ***ClientIP***.
9975990	Non implementata nel simulatore.

7.17.4 SysUDPSktSend, UDP socket send

Type	Library
Function	XTarget_07_0

SysUDPSktSend
File
TxData
Size
PeerIP
PeerPort

Questa funzione esegue la trasmissione dati su di un socket UDP, occorre passare un flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

Nel parametro ***TxData*** bisogna definire l'indirizzo del buffer di memoria che contiene i dati da trasmettere, ed in ***Size*** occorre definire il numero di bytes di dati da trasmettere.

I parametri ***PeerIP*** e ***PeerPort*** indicano l'indirizzo IP e la porta a cui saranno inviati i dati.

Se la trasmissione è riuscita verrà ritornato il numero di bytes trasmessi (Solitamente uguale al valore definito in ***Size***), in caso di errore viene ritornato ***EOF***.

Parametri funzione:

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

TxData (@USINT]) Puntatore al buffer che contiene i dati da trasmettere.

Size (UINT]) Numero di bytes dati da trasmettere.

PeerIP (STRING[15]) Stringa di definizione indirizzo IP a cui trasmettere i dati.

PeerPort (UINT]) Porta a cui trasmettere i dati.

La funzione ritorna:

(INT) Numero bytes trasmessi, ***EOF*** in caso di errore

Codici di errore

In caso di errore la funzione torna con ***EOF***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

Codice	Descrizione
9976005	Blocco funzione non supportato.
9976010	Valore di ***File*** non definito.
9976050	Tipo di ***stream*** definito in ***File*** non corretto.
9976100	Errore indirizzo IP connessioni accettate ***PeerIP***.
9976200	Errore trasmissione dati su socket UDP
9976990	Non implementata nel simulatore.

Esempi

Nell'esempio inviando caratteri al sistema SlimLine sulla porta 1000 da una connessione UDP, i caratteri inviati saranno ritrasmessi sulla porta UDP del sistema client da cui sono stati ricevuti.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RxDone	BOOL	Auto	No	FALSE	..	Rx data done
2	ClientPort	UINT	Auto	No	0	..	Client port
3	RxSize	UINT	Auto	No	0	..	Rx bytes received
4	ClientIP	STRING	Auto	[15]		..	Client IP address
5	PrnDataAdd	DB100AddOffset	Auto	No	0	..	Print data buffer address
6	RxDataAdd	DB100AddOffset	Auto	No	0	..	Rx data buffer address
7	Fp	FILEP	Auto	No		..	File pointer
8	SktLsn	SysSktListen	Auto	No	0	..	FB socket listen
9	SktRcv	SysUDPSktRcv	Auto	No	0	..	FB socket receive

Esempio LD *(PTP119A200, UDPSocketEcho)*

Sysfopen
SysFirstLoop
EN
ENO
'UDPSKT'
FName
Fp
'rw'
Mode

SktLsn
SysSktListen
SysFirstLoop
Enable
Done
Fp
File
Fault
MyIP
Connect
1000
MyPort
ClientIP
LifeTm
ClientPort
FlushTm
RxSize
TxSize

SktRcv
SysUDPSktRcv
RxDone
Enable
Done
Fp
File
Fault
RxDataAdd
'255.255.255.255'
PeerIP
ClientIP
ClientIP
DB100AddOffset
65535
PeerPort
ClientPort
ClientPort
10
Offset
Address
RxData
RxSize
RxSize
10
Size

SysUDPSktSend
RxDone
PrnDataAdd
EN
ENO
DB100AddOffset
Fp
File
TxData
10
Offset
Address
RxSize
Size
ClientIP
PeerIP
ClientPort
PeerPort

7.17.5 UDPDataTxfer, UDP data transfer

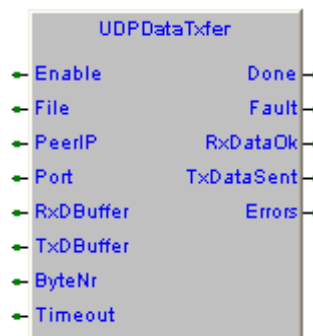
Type	Library
FB	ePLCUtyLib_C000

Questo blocco funzione esegue il trasferimento di un blocco di memoria tra due sistemi utilizzando una connessione UDP su rete ethernet. Occorre passare alla FB un flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen*** ed il socket deve essere stato posto in condizione di listening dalla FB ***SysSktListen***.

Il parametro ***PeerIP*** indica l'indirizzo IP del sistema con cui avviene il trasferimento dati, ***Port*** indica la porta tramite la quale il trasferimento avviene (Deve assumere lo stesso valore su entrambi i sistemi).

Il parametro ***Timeout*** definisce il tempo massimo per il trasferimento dei dati, l'invio dei dati si conclude con la ricezione di un acknowledge da parte dell'altro sistema, un ciclo di invio dati e ricezione acknowledge richiede 2 loop di esecuzione programma. Se dopo l'invio non viene ricevuto Ack entro un tempo pari a ***Timeout/4***, viene effettuato un altro invio e cosi di seguito fino allo scadere del tempo definito. Per garantire almeno 3 retries si consiglia di ***impostare come tempo di timeout un valore pari a 10 volte il tempo massimo di loop*** (Scegliendo quello maggiore tra i due sistemi in comunicazione).

L'invio dei dati è automatico sulla variazione di uno qualsiasi dei bytes del buffer di trasmissione, per garantire il controllo sul collegamento tra i due sistemi, ogni tempo pari a ***Timeout*** viene comunque eseguito un invio del buffer di memoria.

Se i due sistemi sono in comunicazione si attiva l'uscita ***Done***, ***RxDataOk*** si attiva per un loop ad ogni ricezione del buffer dati dall'altro sistema, mentre ***TxDataSent*** si attiva per un loop al termine della trasmissione del buffer dati verso l'altro sistema.

Enable (BOOL)	Abilitazione blocco funzione.
File (FILEP)	Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.
PeerIP (STRING[15])	Stringa di definizione indirizzo IP del sistema con cui avviene il trasferimento dati.
Port (UINT])	Porta tramite la quale avviene il trasferimento dati (Stesso valore su entrambi i sistemi).
RxDBuffer (@USINT])	Puntatore al buffer dove devono essere trasferiti i dati ricevuti.
TxDBuffer (@USINT])	Puntatore al buffer dove sono presenti i dati da trasmettere.
ByteNr (UINT])	Numero di bytes scambiati.
Timeout (UINT])	Tempo massimo per il trasferimento del buffer dati (mS).
Done (BOOL)	Attivo se i due sistemi sono in comunicazione tra di loro.
Fault (BOOL)	Attivo per un loop di programma se errore gestione.
RxDataOk (BOOL)	Attivo per un loop di programma su ricezione buffer dati da altro sistema.
TxDataSent (BOOL)	Attivo per un loop di programma al termine trasmissione buffer dati verso altro sistema.
Errors (UINT])	Numero di errori, incrementato ad ogni nuovo errore, raggiunto valore massimo riparte da 0.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

Codice	Descrizione
10014050	Valore di ***File*** non definito.
10014100	Terminato spazio memoria rilocabile, non è possibile eseguire l''FB.
10014200~2	Errore ricezione frame dati blocco di memoria.
10014300~2	Errore ricezione frame acknowledge.
10014400	Ricevuto comando non gestito.
10014500~1	Errore sequenze di trasmissione.
10014600	Timeout invio frame dati blocco di memoria.

Esempi

Nell'esempio viene scambiato un blocco di memoria di 16 bytes verso il sistema con IP 192.168.0.126.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	PeerIPOk	BOOL	Auto	No	FALSE	..	Peer IP is reached
2	RxData	BOOL	Auto	[0..15]	16(0)	..	Rx data buffer
3	SktAvailable	BOOL	Auto	No	FALSE	..	Socket available
4	TxData	BOOL	Auto	[0..15]	16(0)	..	Tx data buffer
5	Fp	FILEP	Auto	No	0	..	UDP socket
6	IPReach	SysIPReach	Auto	No	0	..	IP reach (Ping)
7	SktLsn	SysSktListen	Auto	No	0	..	Socket listen
8	FBDTxfer	UDPDataTxfer	Auto	No	0	..	UDP data transfer

Esempio LD *(PTP119A300, UDPDataTransfer)*

SysFirstLoop
Sysfopen
EN ENO
'UDPSKT' FName Fp
'rw' Mode

SktLsn
SysSktListen
Enable Done SktAvailable
Fp File Fault
'255.255.255.255' MyIP Connect
128 MyPort ClientIP

IPReach
SysIPReach
Enable Done PeerIPOk
'192.168.0.126' PeerIP Fault
Refresh

FBDTxfer
UDPDataTxsfer
PeerIPOk Enable Done Do00M00
Fp File Fault Do01M00
ADR '192.168.0.126' PeerIP RxDataOk Do02M00
RxData 128 Port TxDataSent Do03M00
RxDBuffer
ADR TxDBuffer
TxData 16 ByteNr
500 Timeout

7.18 Funzioni ed FB supporto prodotti Hw Group (eHwGSpLib)

La ditta della repubblica Ceca Hw Group http://www.hw-group.com produce dispositivi per Networking e prodotti per telecontrollo, monitoraggio e gestione dati.

Tutti i prodotti Hw Group dispongono di connettività su rete ethernet con protocolli TCP/IP, UDP, SNMP, ed per facilitare la connessione di questi prodotti con l'ambiente di sviluppo LogicLab, sono fornite funzioni e blocchi funzioni specifici.

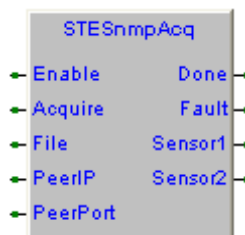
7.18.1 STESnmpAcq, STE termometer acquisition over SNMP

Type	Library
FB	eHwGSpLib_B000

Questo blocco funzione gestisce l'acquisizione del valore di temperatura delle due sonde connesse al termometro IP STE. La connessione tra il termometro e lo SlimLine avviene su rete ethernet utilizzando il protocollo SNMP.

Occorre passare alla FB un flusso dati ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen*** come socket UDP e posto in ascolto con il blocco funzione ***SysSktListen***.

Su fronte attivazione del comando ***Acquire***, viene effettuata la lettura SNMP dal termometro STE definito da indirizzo IP **PeerIP**, su porta ***PeerPort*** (Di default la porta SNMP è la 161). Se il comando ***Acquire*** è mantenuto attivo, la lettura viene effettuata in modo ciclico.

L'uscita ***Done*** si attiva per un loop al termine della acquisizione dei due valori di temperatura.

Enable (BOOL) Abilitazione blocco funzione.

Acquire (BOOL) Comando acquisizione termometro STE. Sul fronte di attivazione si esegue la lettura del valore di temperatura. Mantenuto attivo la lettura viene effettuata in modo ciclico.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

PeerIP (STRING[15]) Stringa di definizione indirizzo IP del termometro IP.

PeerPort (UINT]) Porta utilizzata per la connessione (Di default la porta SNMP è la 161).

Done (BOOL) Attivo per un loop di programma su fine lettura dati.

Fault (BOOL) Attivo per un loop di programma se errore gestione.

Sensor1 (REAL) Valore di temperatura letto dal sensore 1 (°C).

Sensor2 (REAL) Valore di temperatura letto dal sensore 2 (°C).

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10013010 Valore di ***File*** non definito.

10013050 Timeout esecuzione.

10013060 Errore case gestione lettura.

10013100 Errore controllo indirizzo IP del dispositivo STE.

10013120 Errore ricezione dati da dispositivo STE.

10013200~1 Errore lettura sensore 1.

10013300~1 Errore lettura sensore 2.

Esempi

Nell'esempio è gestita l'acquisizione dei due valori di temperatura da un termometro STE ogni secondo. Il valore di temperatura in gradi centigradi è ritornato sulle variabili ***Temperature[0]*** e ***Temperature[1]***. L'uscita logica ***Do00M00*** si attiva alla prima esecuzione della acquisizione, mentre l'uscita logica ***Do01M00*** si attiva in caso di errore di acquisizione.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	SktIsListening	BOOL	Auto	No	FALSE	..	Socket is listening
2	Temperature	REAL	Auto	[0..1]	2(0)	..	Temperature
3	Fp	FILEP	Auto	No	0	..	UDP socket
4	STEAcq	STESnmpAcq	Auto	No	0	..	STE acquisition
5	SktLsn	SysSktListen	Auto	No	0	..	FB socket listen data

Esempio LD *(PTP126A000, LD_STESnmpAcq)*

SysFirstLoop — Sysfopen (EN, ENO; FName: 'UDPSKT'; Mode: 'rw') → Fp

SktLsn — SysSktListen: SysFirstLoop (negated) → Enable; Fp → File; MyIP; 161 → MyPort; LifeTm; FlushTm; RxSize; TxSize; Done → SktIsListening; Fault; Connect; ClientIP; ClientPort

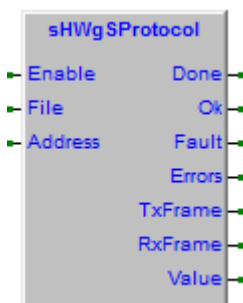
STEAcq — STESnmpAcq: SktIsListening → Enable; SysClock1000 (P) → Acquire; Fp → File; '192.168.0.60' → PeerIP; 161 → PeerPort; Done → Do00M00 (S); Fault → Do01M00 (S); Sensor1 → Temperature[0]; Sensor2 → Temperature[1]

7.18.2 sHWgSProtocol, HW group serial protocol

Type	Library
FB	eHwGSpLib_B000

Questo blocco funzione gestisce la lettura di dispositivi HW group con il protocollo seriale RS485. Occorre passare alla FB il puntatore al file di porta seriale ***stream*** indicato dal parametro ***File***, precedentemente aperto dalla funzione ***Sysfopen***.

Attivando l'ingresso ***Enable*** viene effettuata la lettura del valore dal dispositivo connesso alla porta seriale definita, terminata l'esecuzione si attiva l'uscita ***Done***. Se il comando ha avuto esito positivo si attiva l'uscita ***Ok***, in caso contrario si attiva l'uscita ***Fault***.

Per seguire nuovamente il comando occorre riabilitare l'ingresso ***Enable***, Il blocco funzione è stato realizzato per permetterne la connessione in cascata. In pratica è possibile connettere al ***Done*** di un FB l'***Enable*** di un'altro e così di seguito.

Enable (BOOL) Abilitazione blocco funzione.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

Address (STRING[1]) Stringa di definizione indirizzo dispositivo.

Done (BOOL) Si attiva al termine della esecuzione comando.

Ok (BOOL) Si attiva se acquisizione valore riuscita.

Fault (BOOL) Si attiva su errore gestione.

Errors (UDINT) Numero di errori, incrementato ad ogni nuovo errore, raggiunto valore massimo riparte da 0.

TxFrame (STRING[8]) Contiene il frame inviato al dispositivo, può essere utilizzato in debug.

RxFrame (STRING[16]) Contiene il frame ricevuto dal dispositivo, può essere utilizzato in debug.

Value (REAL) Valore acquisito dal dispositivo.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, il valore di ***Errors*** si incrementa e con ***SysGetLastError*** è possibile rilevare il codice di errore.

10032010 Valore di ***File*** non definito.

10032050 Timeout esecuzione.

10032060 Errore case gestione.

10032100 FB usata in task fast o slow.

10013200~1 Errore nella gestione del protocollo di comunicazione.

10032300 Errore lettura valore.

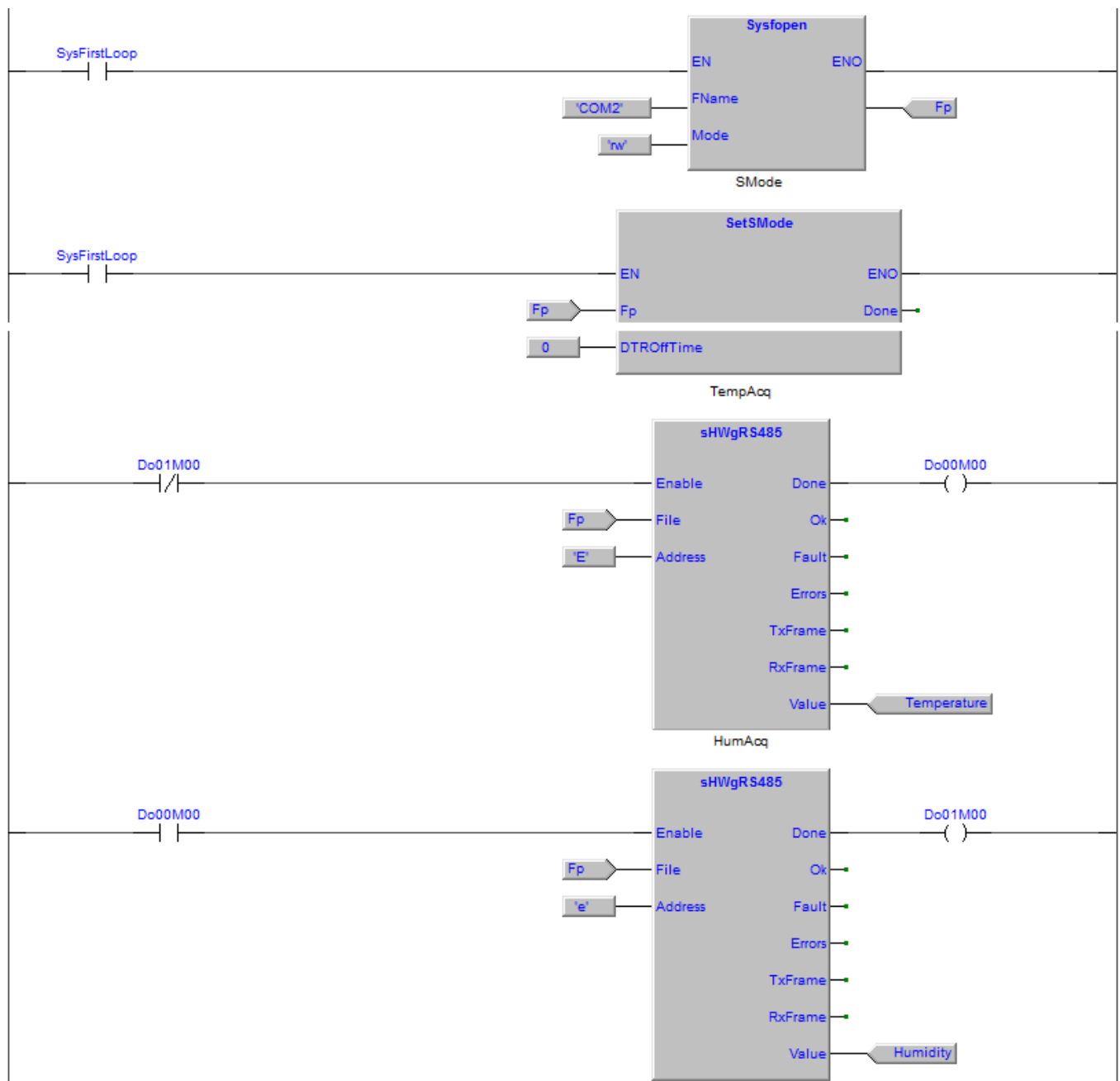
Esempi

Nell'esempio è gestita l'acquisizione del valore di temperatura e di umidità da un HTemp.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Humidity	REAL	Auto	No	0	..	Humidity value (%)
2	Temperature	REAL	Auto	No	0	..	Temperature value (°C)
3	Fp	FILEP	Auto	No	0	..	File pointer
4	SMode	SetSMode	Auto	No	0	..	FB set serial mode data
5	HumAcq	sHWgSProtocol	Auto	No	0	..	FB HTempBox data
6	TempAcq	sHWgSProtocol	Auto	No	0	..	FB HTempBox data

Esempio LD *(PTP126A000, LD_HTBoxRead)*

SysFirstLoop
Sysfopen
EN ENO
'COM2' FName
'rw' Mode
Fp
SMode
SysFirstLoop
SetSMode
EN ENO
Fp Fp Done
0 DTROffTime
TempAcq
Do01M00
sHWgRS485
Enable Done
Do00M00
Fp File Ok
'E' Address Fault
Errors
TxFrame
RxFrame
Value Temperature
HumAcq
Do00M00
sHWgRS485
Enable Done
Do01M00
Fp File Ok
'e' Address Fault
Errors
TxFrame
RxFrame
Value Humidity

7.19 Funzioni ed FB supporto protocollo NMEA (eNMEALib)

Attenzione! Per utilizzare la libreria occorre importarla nel proprio progetto. Vedere capitolo relativo all'import delle librerie.

NMEA 0183 (O più comunemente NMEA) è uno standard di comunicazione di dati utilizzato soprattutto in nautica e nella comunicazione di dati satellitari GPS. L'ente che gestisce e sviluppa il protocollo è la National Marine Electronics Association.

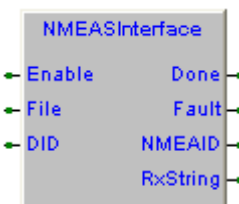
Questo protocollo si basa sul principio che la fonte, detta talker, può soltanto inviare i dati (sentences) e la ricevente, detta listener, può soltanto riceverli.

La libreria ***eNMEALib*** fornisce una serie di funzioni e blocchi funzione per gestire le sentenze NMEA, in pratica è possibile realizzare programmi con l'ambiente di sviluppo LogicLab che si comportano come listeners di sentenze NMEA.

7.19.1 NMEASInterface, NMEA system interface

Type	Library
FB	eNMEALib_B000

Questo blocco funzione gestisce l'interfaccia verso un dispositivo che invia sentenze NMEA connesso al dispositivo di I/O definito in ***File***, questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi ***protezione funzioni e blocchi funzione***. E' comunque possibile utilizzarlo liberamente in modo test per 30 Min.

L'FB riceve le sentenze NMEA dal dispositivo, ne controlla il prefisso comparandolo con la stringa definita in ***DID***, controlla se la sentenza ricevuta è corretta (Controllo sul CRC). L'uscita ***Done*** si attiva per un loop ad ogni ricezione di sentenza NMEA corretta.

L'FB ritorna un ***NMEAID*** che deve essere passato alle FB collegate (FB di gestione sentenze NMEA). L'uscita ***RxString*** riportano la stringa ricevuta dal dispositivo, in questo modo è possibile visualizzare in debug la comunicazione con il dispositivo permettendo di visualizzare eventuali errori.

Enable (BOOL) Abilitazione blocco funzione.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

DID (STRING[2]) Stringa di definizione prefisso dispositivo.

Done (BOOL) Attivo per un loop di programma su ricezione sentenza corretta.

Fault (BOOL) Attivo per un loop di programma se errore gestione.

NMEAID (UDINT) ID interfaccia con sistema NMEA da passare alle FB collegate.

RxString (STRING[82]) Contiene la stringa ricevuta dal dispositivo, può essere utilizzato in debug per verificare le sentenze ricevute.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10017010 Valore di ***File*** non definito.

10017020 FB protetta, terminato tempo funzionamento in modo demo.

10017050 Timeout esecuzione.

10017070 Errore case gestione.

10017100~4 Errore ricezione sentenza NMEA.

7.19.2 GLLSentence, Geographic Position sentence

Type	Library
FB	eNMEALib_B000

Questo blocco funzione esegue la ricezione della sentenza GLL Geographic Position, si collega al blocco funzione ***NMEASInterface*** di gestione dispositivo NMEA. Occorre passare ***NMEAID*** in uscita dal blocco funzione di gestione dispositivo.

La sentenza GLL contiene le informazioni di latitudine, longitudine ora e fix, esempio di sentenza ***$IIGLL,4419.0173,N,00829.6653,E,084550.00,A,2*09***.

L'FB controlla correttezza dei campi della sentenza e ne estrae le informazioni di latitudine, longitudine e tempo. L'uscita ***Done*** si attiva per un loop ad ogni ricezione di sentenza GLL corretta.

Enable (BOOL) Abilitazione blocco funzione.

NMEAID (UDINT) ID interfaccia con sistema NMEA fornito in uscita dal blocco funzione ***NMEASInterface***.

Done (BOOL) Attivo per un loop di programma su ricezione sentenza GLL corretta.

Fault (BOOL) Attivo per un loop di programma se errore sentenza.

Latitude (REAL) Valore di latitudine indicato nella sentenza, il valore è espresso in frazione di gradi. Valori positivi indicano latitudine nord, valori negativi latitudine sud.

Longitude (REAL) Valore di longitudine indicato nella sentenza, il valore è espresso in frazione di gradi. Valori positivi indicano longitudine est, valori negativi longitudine ovest.

UTCHour (USINT) Valore di ora UTC indicato nella sentenza.

UTCMinute (USINT) Valore di minuti UTC indicato nella sentenza.

UTCSecond (USINT) Valore di secondi UTC indicato nella sentenza.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10018010 ***NMEAID*** non definito.

10018020 ***NMEAID*** non corretto.

10018100~2 Errore nel valore di latitudine.

10018200~2 Errore nel valore di longitudine.

10018300~2 Errore nel valore ora UTC.

Esempi

E' disponibile un programma di esempio Ptp123*000 che gestisce l'interfaccia verso un navigatore satellitare con l'interpretazione di alcune sentenze NMEA.

Nell'esempio riportato è gestita la ricezione di una sentenza GLL.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	GLL	GLLSentence	Auto	No	0	..	GL sentence reception FB data
3	NMEAID	UDINT	Auto	No	0	..	NMEA interface ID
4	NMEARx	NMEASInterfa	Auto	No	0	..	NMEA device interface FB data
5	RxGPS	STRING	Auto	[32]		..	Rx data from GPS device
6	SMode	SetSMode	Auto	No	0	..	Serial mode FB data

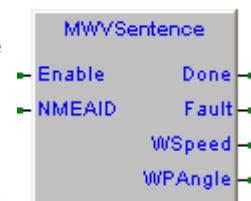
Esempio LD

Type	Library
FB	eNMEALib_B000

7.19.3 MWVSentence, Wind Speed and Angle sentence

Questo blocco funzione esegue la ricezione della sentenza MWV wind speed and angle, si collega al blocco funzione ***NMEASInterface*** di gestione dispositivo NMEA. Occorre passare ***NMEAID*** in uscita dal blocco funzione di gestione dispositivo.

La sentenza MWV contiene le informazioni di velocità e direzione del vento, esempio di sentenza ***$IIMWV,120.09,R,4.53,N,A*35***.

L'FB controlla correttezza dei campi della sentenza e ne estrae le informazioni di velocità e direzione. L'uscita ***Done*** si attiva per un loop ad ogni ricezione di sentenza MWv corretta.

Enable (BOOL) Abilitazione blocco funzione.

NMEAID (UDINT) ID interfaccia con sistema NMEA fornito in uscita dal blocco funzione ***NMEASInterface***.

Done (BOOL) Attivo per un loop di programma su ricezione sentenza MWV corretta.

Fault (BOOL) Attivo per un loop di programma se errore sentenza.

WSpeed (REAL) Valore di velocità del vento (Nodi).

WPAngle (REAL) Valore angolo polare (Relative) direzione vento.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10020010 ***NMEAID*** non definito.

10020020 ***NMEAID*** non corretto.

10020100 Errore nel valore di velocità vento.

7.20 Funzioni ed FB supporto inverter Power One (ePowerOneLib)

Power One è uno dei principali produttori mondiale di sistemi di alimentazione. Power One è di diritto anche nel settore delle energie alternative con applicazioni per sistemi eolici ed inverter fotovoltaici. Oggi una strategia convincente nell'ambito delle energie alternative non può prescindere dallo sviluppo di soluzioni per il risparmio energetico.

La linea di Inverter fotovoltaici Aurora, comprende sia modelli per la connessione in rete sia isolati, con o senza trasformatore e concepiti per applicazioni da esterno e da interno. Tutti i prodotti della gamma si posizionano per soluzioni di progetto e tecnologia costruttiva ai vertici del mercato e sono caratterizzati da elevatissima affidabilità, innovazione ed efficienza.

Inverter Aurora

Alta efficienza di conversione e estrema facilità di manutenzione grazie alla possibilità di inserzione e disinserzione rapida dei moduli fotovoltaici. La scalabilità del sistema grazie alla architettura "Add-on" consente di coprire un'ampia gamma di applicazioni (fino a 300kW su singolo armadio).

Disponibile anche la versione senza trasformatore BT per connessione diretta ad una cella di media tensione (con trasf. MT).

7.20.1 AuroraDSPMeasure, Aurora measure request to DSP

Type	Library
FB	ePowerOneLib_B000

Questo blocco funzione esegue la lettura delle misure dal DSP di un inverter Aurora della Power One, connesso al dispositivo di I/O definito in ***File***. Questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi ***protezione funzioni e blocchi funzione***. E' comunque possibile utilizzarlo liberamente in modo test per 30 Min. Viene utilizzata la funzione ***GetPolynomialCRC*** per il calcolo del CRC dei frame dati da e verso l'inverter.

La connessione con gli inverters è in RS485 multidrop, occorre definire in ***Address*** l'indirizzo dell'inverter con cui si vuole dialogare. In ***Measure*** occorre indicare il codice della misura da leggere (Vedi codici misura).

Attivando l'ingresso Start viene eseguita la lettura della misura indicata, terminata la lettura viene attivata per un loop l'uscita ***Done***, in caso di errore esecuzione viene attivata per un loop l'uscita ***Fault*** ed incrementato il valore in ***Errors***.

Enable (BOOL) Comando di abilitazione blocco funzione.

Start (BOOL) Comando di esecuzione lettura misura.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

Address (USINT) Indirizzo inverter (Range da 0 a 255).

Measure (USINT) Tipo misura da effettuare su inverter (Vedi codici misura).

Done (BOOL) Attivo per un loop al termine della esecuzione del comando.

Fault (BOOL) Attivo per un loop su errore esecuzione del comando.

Value (REAL) Valore misura acquisito da inverter (E' nella relativa unità di misura).

Errors (UDINT) Numero di errori, incrementato ad ogni nuovo errore, raggiunto valore massimo riparte da 0.

Codici misura

Nella variabile ***Measure*** occorre definire il codice della misura da effettuare dall'inverter secondo la tabella.

Codice	Descrizione	Um
1	Grid Voltage (For three-phases systems is the mean)	V
2	Grid Current (For three-phases systems is the mean)	A
3	Grid Power (For three-phases systems is the mean)	W
4	Frequency (For three-phases systems is the mean)	Hz
5	Vbulk (For Inverter with more Bulk is the sum)	V
6	Ileak (Dc/Dc)	A
7	Ileak (Inverter)	A
21	Inverter Temperature	°C
22	Booster Temperature	°C
23	Input 1 Voltage (Input Voltage for single channel module)	V
25	Input 1 Current (Input Current for single channel module)	A
26	Input 2 Voltage (Input Voltage for single channel module)	V
27	Input 2 Current (Input Current for single channel module)	A
28	Grid Voltage (Dc/Dc)	V
29	Grid Frequency (Dc/Dc)	Hz

Codice	Descrizione	Um
30	Isolation Resistance (Riso)	
31	Vbulk (Dc/Dc)	V
32	Average Grid Voltage (VgridAvg)	V
33	VbulkMid	V
34	Power Peak	W
35	Power Peak Today	W
36	Grid Voltage neutral	V
37	Wind Generator Frequency	Hz
38	Grid Voltage neutral-phase	V
39	Grid Current phase r	A
40	Grid Current phase s	A
41	Grid Current phase t	A
42	Frequency phase r	Hz
43	Frequency phase s	Hz
44	Frequency phase t	Hz
45	Vbulk +	V
46	Vbulk -	V
47	Supervisor Temperature	°C
48	Alim. Temperature	°C
49	Heat Sink Temperature	°C
61	Grid Voltage phase r	V
62	Grid Voltage phase s	V
63	Grid Voltage phase t	V

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10030010 Valore di ***File*** non definito.

10030020 FB protetta, terminato tempo funzionamento in modo demo.

10030050 Timeout esecuzione.

10030070 Errore case gestione.

10030100 Errore CRC risposta da inverter Aurora.

10030200 Errore ricezione “Transmission state” da inverter Aurora.

10030251 Errore da inverter Aurora “Command is not implemented”.

10030252 Errore da inverter Aurora “Variable does not exist”.

10030253 Errore da inverter Aurora “Variable value is out of range”.

10030254 Errore da inverter Aurora “EEprom not accessible”.

10030255 Errore da inverter Aurora “Not Toggled Service Mode”.

10030256 Errore da inverter Aurora “Can not send the command to internal micro”.

10030257 Errore da inverter Aurora “Command not Executed”.

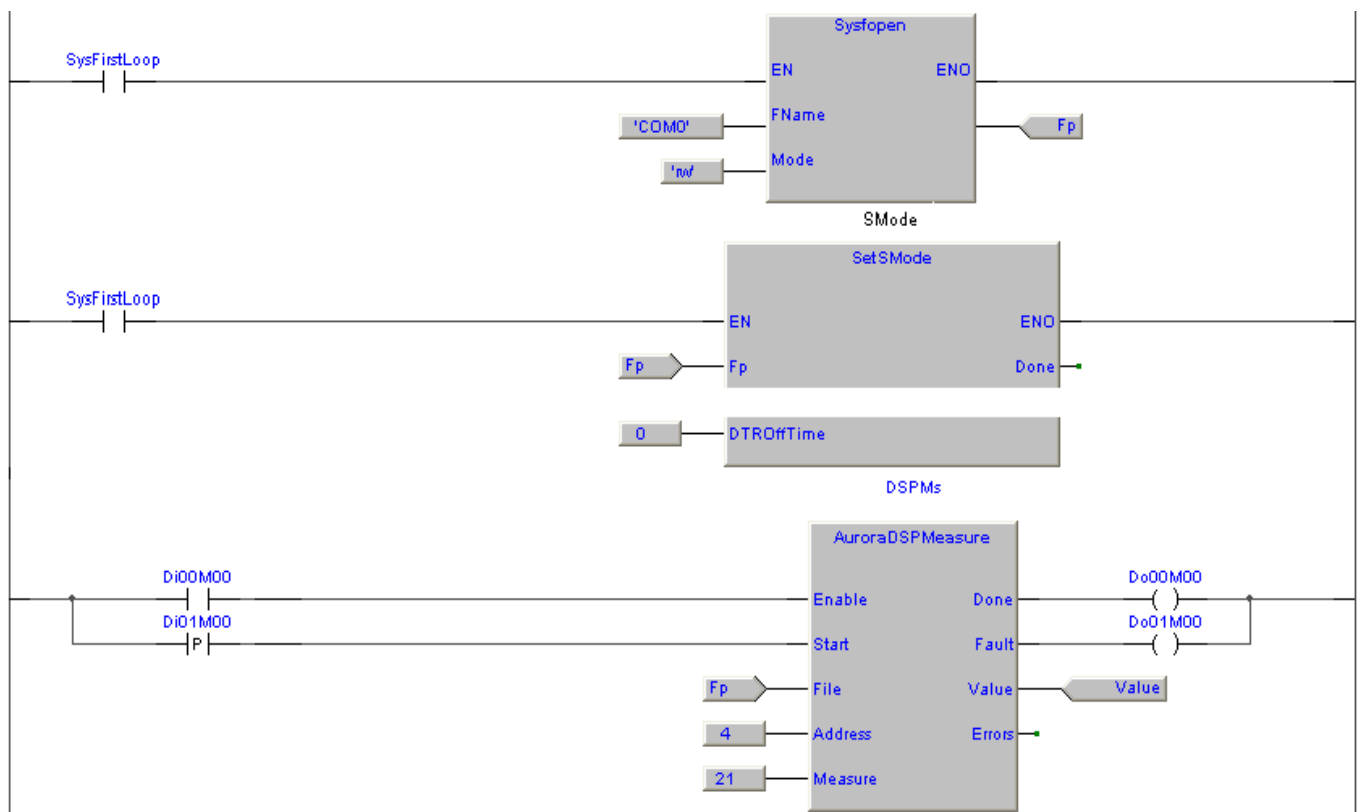
10030258 Errore da inverter Aurora “The variable is not available, retry”.

Esempi

Viene eseguita la lettura della misura 21 (Inverter Temperature) dall'inverter con indirizzo 4, il valore ritornato è trasferito nella variabile ***Value***. Di default la porta seriale và impostata a ***19200, n 8, 1***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Terminal I/O file pointer
2	SMode	SetSMode	Auto	No	0	..	FB set serial mode
3	DSPMs	AuroraDSPMeasure	Auto	No	0	..	FB Aurora DSP measure
4	Value	REAL	Auto	No	0	..	Value read from inveter

Esempio LD

7.20.2 AuroraCEnergy, Aurora cumulated energy reading

Type	Library
FB	ePowerOneLib_B000

Questo blocco funzione esegue la lettura della energia generata da un inverter Aurora della Power One, connesso al dispositivo di I/O definito in ***File***. Questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi ***protezione funzioni e blocchi funzione***. E' comunque possibile utilizzarlo liberamente in modo test per 30 Min. Viene utilizzata la funzione ***GetPolynomialCRC*** per il calcolo del CRC dei frame dati da e verso l'inverter.

La connessione con gli inverters è in RS485 multidrop, occorre definire in ***Address*** l'indirizzo dell'inverter con cui si vuole dialogare. In ***Parameter*** occorre indicare il parametro da leggere (Vedi codici parametro).

Attivando l'ingresso Start viene eseguita la lettura della misura indicata, terminata la lettura viene attivata per un loop l'uscita ***Done***, in caso di errore esecuzione viene attivata per un loop l'uscita ***Fault*** ed incrementato il valore in ***Errors***.

Enable (BOOL) Comando di abilitazione blocco funzione.

Start (BOOL) Comando di esecuzione lettura misura.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

Address (USINT) Indirizzo inverter (Range da 0 a 255).

Parameter (USINT) Codice parametro da acquisire da inverter (Vedi codici parametro).

Done (BOOL) Attivo per un loop al termine della esecuzione del comando.

Fault (BOOL) Attivo per un loop su errore esecuzione del comando.

Value (UDINT) Valore parametro acquisito da inverter (E' nella relativa unità di misura).

Errors (UDINT) Numero di errori, incrementato ad ogni nuovo errore, raggiunto valore massimo riparte da 0.

Codici parametro

Nella variabile ***Parameter*** occorre definire il codice del parametro da leggere dall'inverter secondo la tabella.

Codice	Descrizione	Um
0	Daily energy	Kw
1	Weekly Energy	Kw
3	Month Energy (Energy from the first day of current calendar month)	Kw
4	Year Energy (Energy from the first day of current calendar year)	Kw
5	Total Energy (Total lifetime)	Kw
6	Partial Energy (Cumulated since reset)	Kw

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10031010 Valore di ***File*** non definito.

10031020 FB protetta, terminato tempo funzionamento in modo demo.

10031050 Timeout esecuzione.

10031060 Codice parametro errato.

10031070 Errore case gestione.

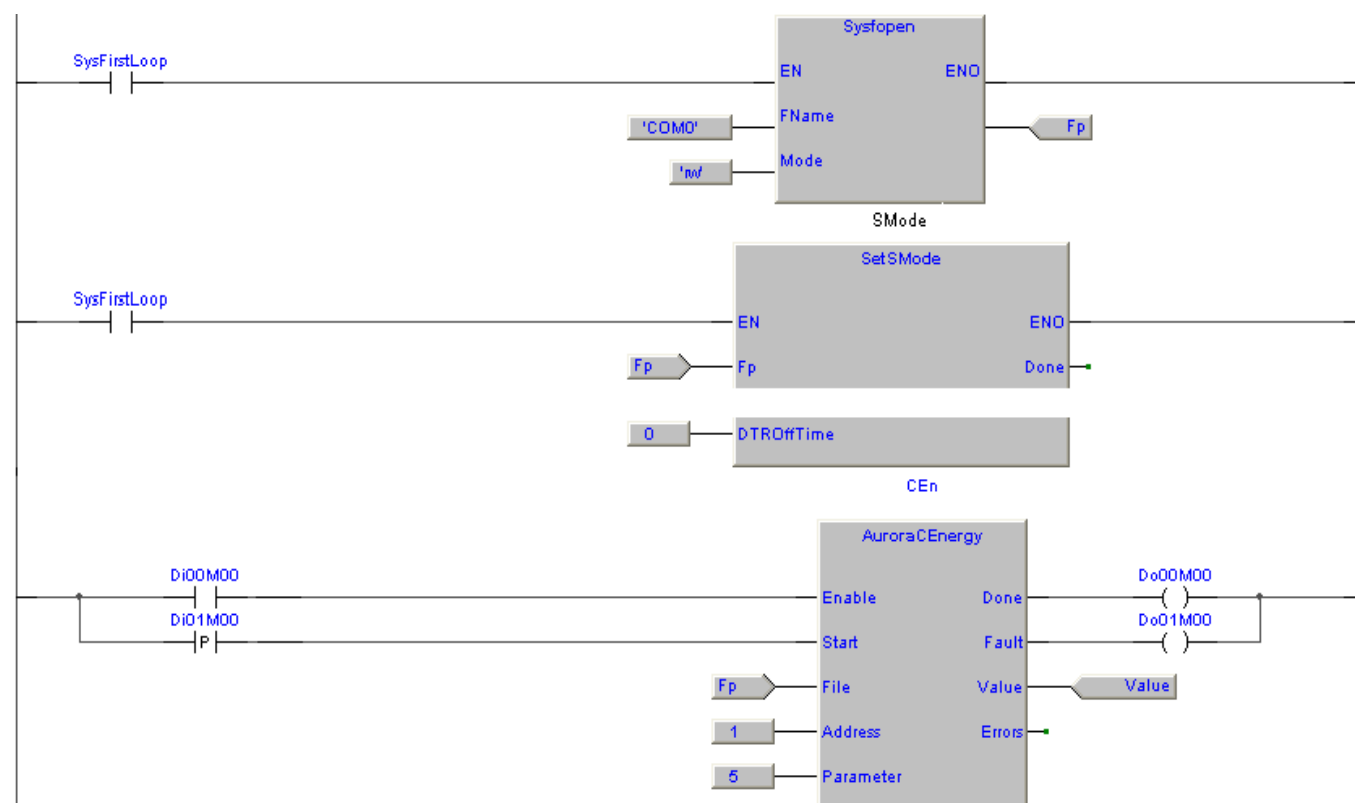
10031100 Errore CRC risposta da inverter Aurora.

Esempi

Viene eseguita la lettura del totale energia prodotta dall'inverter con indirizzo 1, il valore ritornato è trasferito nella variabile ***Value***. Di default la porta seriale và impostata a ***19200, n 8, 1***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	Terminal I/O file pointer
2	SMode	SetSMode	Auto	No	0	..	FB set serial mode
3	Value	UDINT	Auto	No	0	..	Value read from inveter
4	CEn	AuroraCEnergy	Auto	No	0	..	FB Aurora cumulated energy

Esempio LD

7.21 Funzioni ed FB supporto log (eLogLib)

Questa libreria rende disponibili una serie di funzioni e di blocchi funzione per la gestione di logs.

Gestione invio notifiche a server Syslog

SYSLOG (System Log) è un protocollo appartenente alla Suite di protocolli Internet utilizzato per trasmettere attraverso una rete semplici informazioni di log. Il protocollo è stato standardizzato dall' IETF.

Generalmente viene utilizzato via UDP attraverso la porta 514; in particolari applicazioni dove il monitoraggio è fondamentale oppure certi eventi possono innescare azioni da parte del server SYSLOG, si ricorre ad implementazioni TCP e/o a crittografia.

Il client invia un certo messaggio di testo, al massimo 1024 caratteri, al server, comunemente definito come "syslogd", "syslog daemon" o "syslog server". La semplicità del protocollo fa sì che il server possa gestire messaggi provenienti da una variegata tipologia di macchine, da computer, stampanti, dispositivi di rete, macchinari, ecc. Il server può limitarsi a registrare l'evento, per avere un archivio centrale degli avvenimenti, oppure reagire a particolari livelli di severità chiamando programmi, inviando e-mail, ecc. ecc.

Un messaggio di notifica inviato al server Syslog inizia con un indicatore di impianto ***Facility*** e di gravità ***Severity***. Di seguito tabelle con indicazione dei codici assegnati.

	Facility codes
0	Kernel messages
1	User-level messages
2	Mail system
3	System daemons
4	Security/authorization messages
5	Messages generated internally by syslogd
6	Line printer subsystem
7	Network news subsystem
8	UUCP subsystem
9	Clock daemon
10	Security/authorization messages
11	FTP daemon
12	NTP subsystem

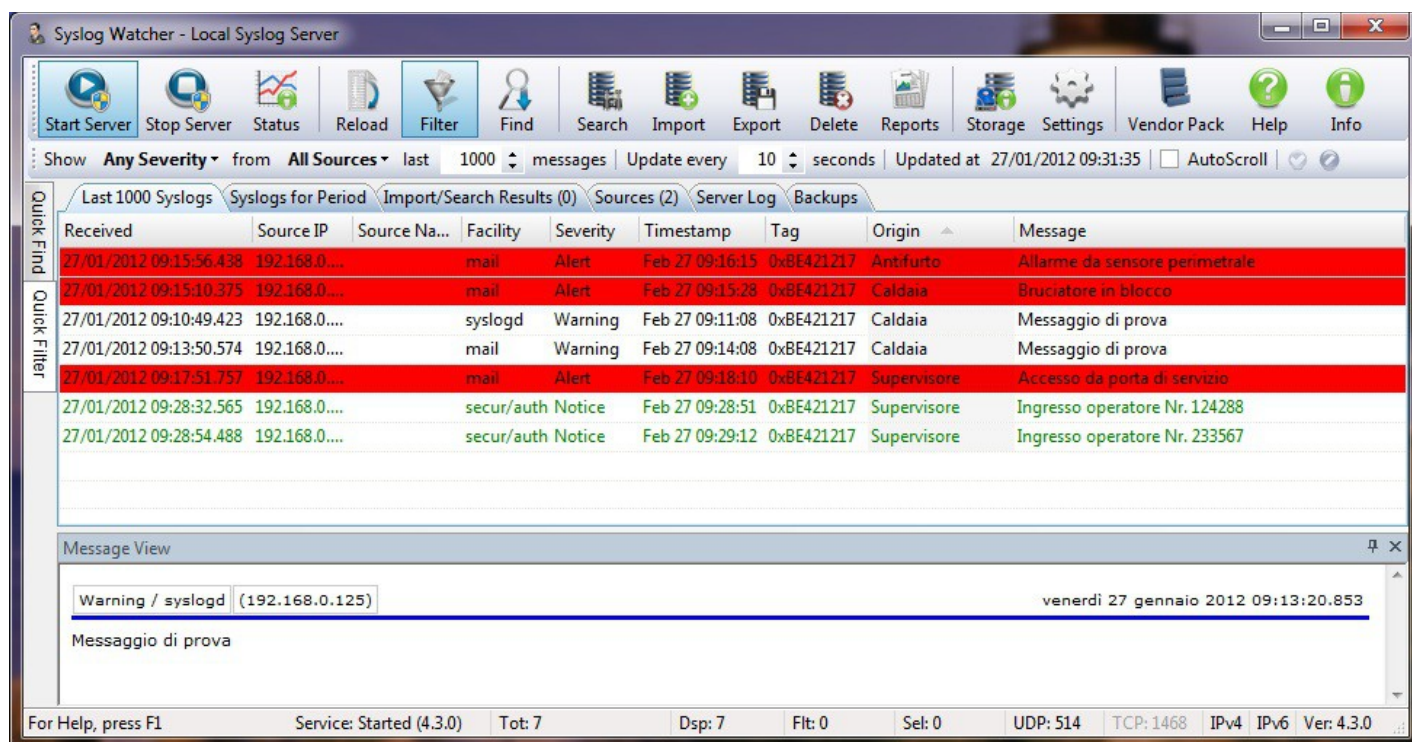
	Severity codes
0	Emergencies Sistema inutilizzabile
1	Alerts Richiede intervento immediato
2	Critical Condizioni critiche
3	Errors Condizione d'errore
4	Warnings Condizioni di warning
5	Notifications Condizioni di anomalia non critici (bugs)
6	Informational Messaggi informativi
7	Debugging Messaggi di debug

Il messaggio prosegue poi con l'indicazione di data e ora, del nome del dispositivo che ha inviato il messaggio ***Hostname*** ed il testo del messaggio ***Message***.

Server Syslog

Un server SYSLOG è un punto centrale dove far arrivare tutti i messaggi di errore dei vari apparati hardware e software di una rete quali router, switch, server, stampanti ecc... per avere un controllo centralizzato degli errori degli apparati.

SYSLOG è particolarmente diffuso in unix e conseguentemente sotto linux, in ambiente windows esistono programmi freeware e/o a pagamento per la gestione del server. Il server può selezionare (filtrare) i messaggi in arrivo in base a diversi criteri, ad ogni selezione corrisponde almeno una azione.

In pratica si tratta di stabilire dei criteri di selezione e l'azione da intraprendere in funzione della provenienza e tipo di messaggio

Il criterio di selezione può essere ad esempio:

Priorità, Indirizzo IP del mittente del messaggio, Nome del Host, Testo (o parte di esso) del messaggio, Intervallo di tempo di uno o più giorni della settimana.

Le azioni possono essere ad esempio:

Nessuna (ignorare il messaggio), Visualizzarlo nel programma di monitoraggio, Inviarlo ad un altro Syslog server, Emettere un suono di allarme, Eseguire un programma, Inviare un E-mail, Memorizzare il messaggio in un Database (esempio MySQL), Memorizzare il messaggio in un logfile, Eseguire uno script, ecc...

Per le mie necessità ho utilizzato ***Syslog Watcher*** (Sito http://www.snmpsoft.com), un ottimo programma gratuito, di seguito uno screenshot con la visualizzazione di notifiche inviate da un sistema SlimLine.

Come si vede le varie notifiche sono suddivise con colori in base alla loro importanza e nel testo del messaggio è possibile riportare qualsiasi informazione ad esempio il codice operatore che ha avuto accesso all'ingresso rilevato da un lettore RFID.

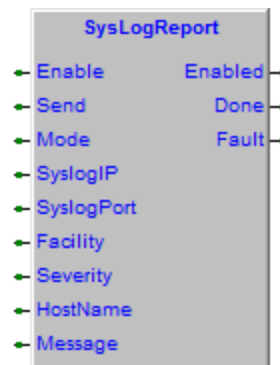
7.21.1 SysLogReport, send a report to Syslog server

Type	Library
FB	eLogLib_B000

Questo blocco funzione esegue l'invio di messaggi di notifica ad un server Syslog il cui indirizzo IP è definito in ***SyslogIP*** e la porta in ***SyslogPort***. Questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi ***protezione funzioni e blocchi funzione***. E' comunque possibile utilizzarlo liberamente in modo test per 30 Min.

E' possibile impostare sia il codice di impianto ***Facility*** che di gravità ***Severity*** oltre al nome del sistema host ***Hostname***.

Attivando l'ingresso Send viene inviata la notifica al server Syslog, eseguito l'invio viene attivata per un loop l'uscita ***Done***, in caso di errore esecuzione viene attivata per un loop l'uscita ***Fault***.

SysLogReport
Enable — Enabled
Send — Done
Mode — Fault
SyslogIP
SyslogPort
Facility
Severity
HostName
Message

Enable (BOOL) Comando di abilitazione blocco funzione.

Send (BOOL) Comando di invio notifica.

Mode (USINT) Modo operativo, 0:UDP.

SyslogIP (STRING[16]) Indirizzo IP server Syslog

SyslogPort (UINT) Porta utilizzata dal server Syslog.

Facility (USINT) Codice impianto.

Severity (USINT) Codice gravità.

HostName (STRING[32]) Nome sistema che invia il messaggio di notifica.

Message (STRING[160]) Testo descrittivo del messaggio di notifica.

Enabled (BOOL) Attivo su abilitazione blocco funzione.

Done (BOOL) Attivo per un loop al termine dell'invio del messaggio di notifica.

Fault (BOOL) Attivo per un loop su errore esecuzione del comando.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10034020 FB protetta, terminato tempo funzionamento in modo demo.

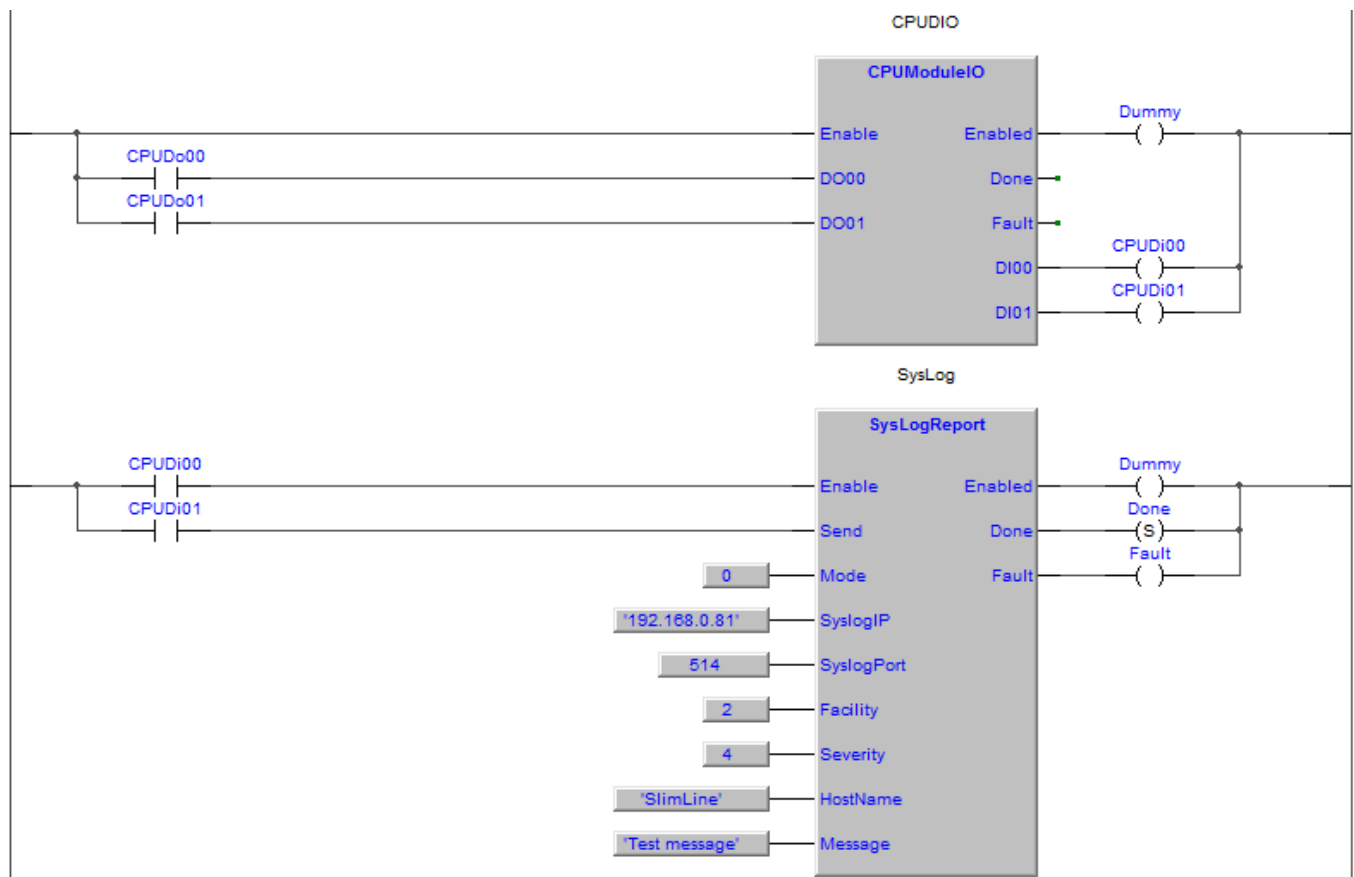
10034100 Server Syslog non raggiungibile, il server non risponde al ping.

Esempi

Nel seguente esempio sono acquisiti gli I/O del modulo CPU, utilizzando i due ingressi è possibile eseguire l'invio di un messaggio di notifica al server Syslog con IP 192.168.0.81 sulla porta 514 in UDP.

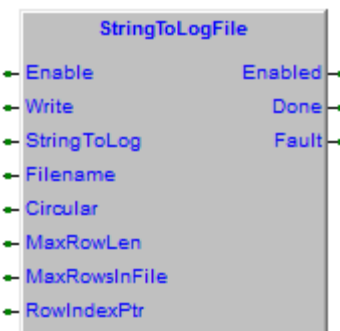
Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Done	BOOL	Auto	No	FALSE	..	Syslog sent
2	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
3	Fault	BOOL	Auto	No	FALSE	..	Syslog error
4	CPUDIO	CPUModuleIO	Auto	No	0	..	FB CPU I/O management
5	SysLog	SysLogReport	Auto	No	0	..	FB SysLogReport

Esempio LD

7.21.2 StringToLogFile, salva una stringa in un file di log

Type	Library
FB	eLogLib_B000

Questo blocco funzione effettua il log della stringa ***StringToLog*** nel file su disco di nome ***Filename***. Ogni riga viene terminata con CR-LF. Al raggiungimento di ***MaxRowsInFile*** righe nel file, se ***Circular*** è settato, si riprende a salvare dall'inizio del file sovrascrivendo le righe già presenti. Se ***Circular*** non è settato non viene più effettuato il log segnalando errore. La variabile ***MaxRowsLen*** permette di limitare la lunghezza massima di ciascuna riga presente nel file di log.

Se ***Circular*** è settato se la lunghezza di ***StringToLog*** è inferiore a ***MaxRowsLen***, vengono aggiunti degli spazi sino ad avere lunghezza definita. In ***RowIndexPtr*** occorre inserire l'indirizzo della variabile usata come indice della successiva riga in cui fare log. Normalmente deve essere una variabile tamponata per permettere di scrivere nel punto giusto anche dopo uno spegnimento-accensione del sistema. Se invece non si usa una variabile tamponata, ad ogni spegnimento-accensione, si ripartirà a scrivere dall'inizio del file. Quindi si può scegliere il comportamento voluto.

StringToLogFile
Enable — Enabled
Write — Done
StringToLog — Fault
Filename
Circular
MaxRowLen
MaxRowsInFile
RowIndexPtr

Questo è un blocco funzione protetto. Per utilizzarlo occorre richiedere il codice di protezione, vedi ***protezione funzioni e blocchi funzione***. E' comunque possibile utilizzarlo liberamente in modo test per 30 Min.

Attivando l'ingresso ***Write*** viene salvata su disco la stringa ***StringToLog***. Eseguito il salvataggio, viene attivata per un loop l'uscita ***Done***. In caso di errore esecuzione viene attivata per un loop l'uscita ***Fault***.

Enable (BOOL)	Comando di abilitazione blocco funzione.
Write (BOOL)	Comando di scrittura di ***StringToLog*** nel file.
StringToLog (STRING[160])	Stringa di cui fare log.
Filename (STRING[32])	Percorso e nome del file in cui fare log (es.: 'SDCard/MyFile.txt').
Circular (BOOL)	Indica se effettuare il riporto circolare delle righe di log.
MaxRowLen (USINT)	Numero massimo di caratteri costituenti la singola riga del file di log.
MaxRowsInFile (UDINT)	Numero massimo di righe nel file.
RowIndexPtr (@UDINT)	Pointer alla variabile usata come indice della successiva riga di log.
Enabled (BOOL)	Attivo su abilitazione blocco funzione.
Done (BOOL)	Attivo per un loop se la scrittura nel file è riuscita.
Fault (BOOL)	Attivo per un loop su errore esecuzione del comando.

Codici di errore

In caso di errore si attiva l'uscita ***Fault*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

10035020 FB protetta. Terminato il tempo di funzionamento in modo demo.

10035100 Il valore di ***MaxRowLen*** è troppo grande.

10035110 Raggiunto il massimo numero di righe di log.

10035120 Errore apertura file.

10035130 Errore posizionamento nel file.

10035140 Errore scrittura nel file.

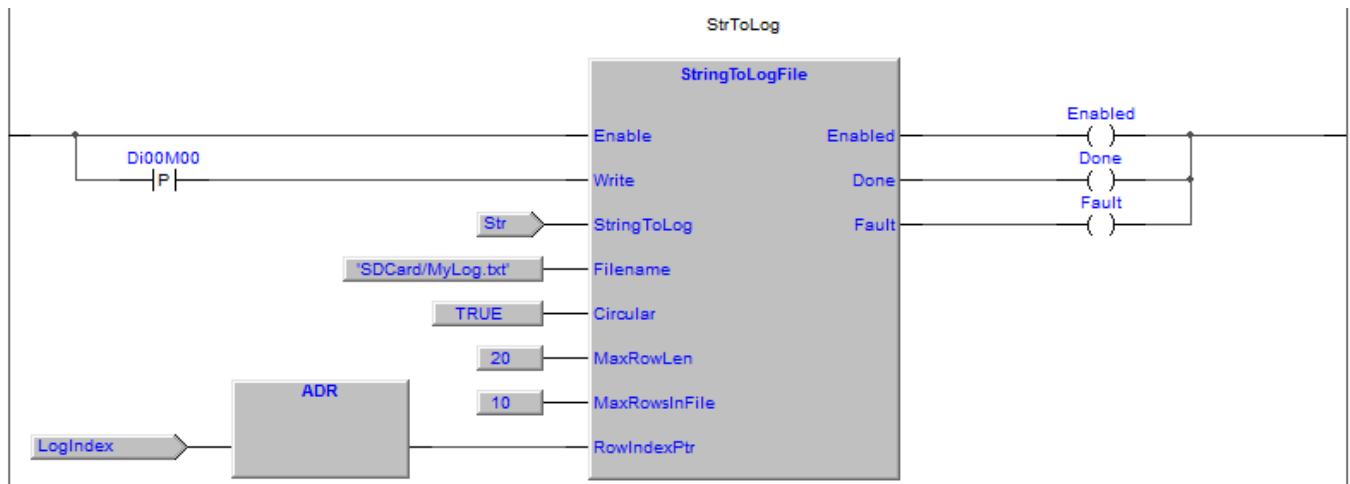
10035150 Errore chiusura file.

Esempi

Nel seguente esempio ad ogni attivazione dell'input ***Di00M00***, viene salvata la stringa presente in Str nel file di nome ***SDCard/MyLog.txt***. Avendo impostato ***Circular*** uguale ***TRUE***, al raggiungimento di 10 log, l'undicesimo sovrascrive il primo. La variabile ***LogIndex*** è una variabile mappata nell'area delle variabili tamponate.

Definizione variabili

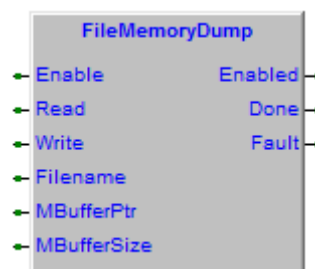
	Name	Type	Address	Array	Init value	Attribute	Description
1	Str	STRING	Auto	[32]	Log string	..	String to log
2	StrToLog	StringToLogFile	Auto	No	0	..	FB String to log file
3	Done	BOOL	Auto	No	FALSE	..	FB done
4	Fault	BOOL	Auto	No	FALSE	..	FB fault
5	Enabled	BOOL	Auto	No	FALSE	..	FB enabled

Esempio LD

7.21.3 FileMemoryDump, dump memoria su file

Type	Library
FB	eLogLib_B000

Questo blocco funzione effettua il dump di una zona di memoria memoria a partire da indirizzo definito in ***MBufferPtr*** per il numero di bytes definito in ***MBufferSize*** su di un file su disco di nome ***Filename***.

Con un comando impulsivo su ingresso ***Write*** viene creato il file ed il contenuto della memoria viene scritto nel file. Con un comando impulsivo su ingresso ***Read*** viene letto il file ed il suo contenuto trasferito in memoria.

Terminato il comando viene attivata per un loop l'uscita ***Done***, in caso di errore esecuzione viene attivata per un loop l'uscita ***Fault***.

Il file di dump su disco è un file ascii quindi è possibile editarlo con un qualsiasi text editor, ecco un esempio di file.

```
00000000: 00 AB 12 34 00 00 00 00 | 00 00 00 12 00 0F 0A CC
00000010: 02 00 00 00 00 00 00 00 | EF C0 DD 00 00 00 01 00
```

Enable (BOOL) Comando di abilitazione blocco funzione.

Read (BOOL) Comando di lettura da dump file in memoria.

Write (BOOL) Comando di scrittura memoria su dump file.

Filename (STRING[32]) Percorso e nome del file in cui fare log (es.: 'Storage/Dump.txt').

MBufferPtr (@USINT) Pointer a buffer di memoria su cui opera il dump.

MBufferSize (UDINT) Dimensione in bytes del buffer di memoria.

Enabled (BOOL) Attivo su abilitazione blocco funzione.

Done (BOOL) Attivo per un loop se comando eseguito.

Fault (BOOL) Attivo per un loop su errore esecuzione del comando.

Codici di errore

In caso di errore si attiva l'uscita ***Fault*** e con ***SysGetLastError*** è possibile rilevare il codice di errore.

10036070 Errore case gestione.

10036100 FB usata in task fast o slow.

10036200 Errore apertura file su comando di lettura.

10036210 Errore posizionamento file su comando di lettura.

10036220 Errore lettura da file su comando di lettura.

10036230 Errore chiusura file su comando di lettura.

10036240~4 Errore dati presenti su file di dump su comando di lettura.

10036400 Errore apertura file su comando di scrittura.

10036410 Errore posizionamento file su comando di scrittura.

10036420 Errore scrittura su file su comando di scrittura.

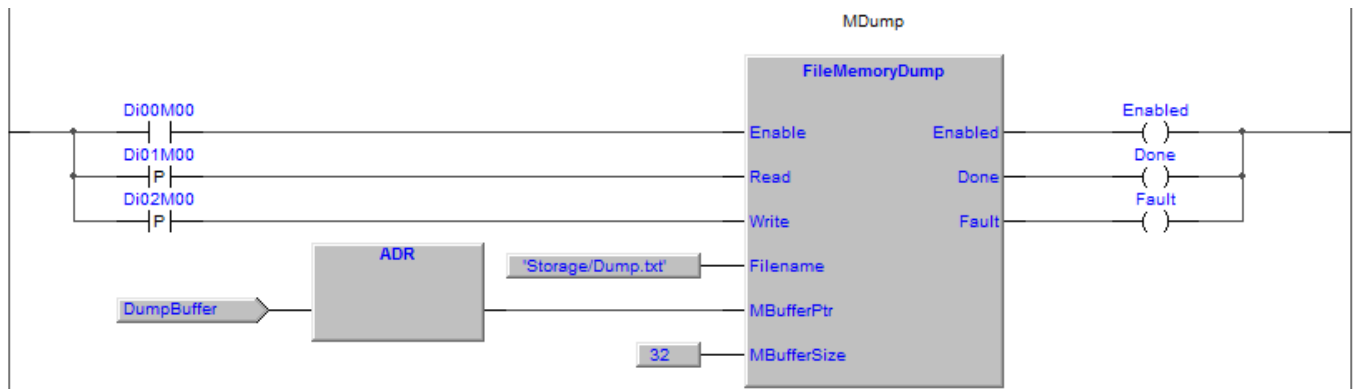
10036430 Errore chiusura file su comando di scrittura.

Esempi

Nel seguente esempio sul fronte di attivazione dell'input ***Di02M00*** viene eseguito il dump del buffer di memoria ***DumpBuffer*** nel file ***Storage/Dump.txt***. Dopo aver eseguito il dump attivando l'input ***Di01M00*** è possibile rileggere i dati dal file di dump e ritrasferirli nel buffer di memoria.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	MDump	FileMemoryDump	Auto	No	0	..	FB File memory dump
2	DumpBuffer	USINT	Auto	[0..31]	32(0)	..	Data buffer
3	Enabled	BOOL	Auto	No	FALSE	..	FB enabled
4	Done	BOOL	Auto	No	FALSE	..	FB done
5	Fault	BOOL	Auto	No	FALSE	..	FB fault

Esempio LD

7.22 Funzioni ed FB comunicazione multimaster (eMMasterDTxferLib)

Attenzione! Per utilizzare la libreria occorre importarla nel proprio progetto. Vedere capitolo relativo all'import delle librerie.

Questa libreria rende disponibili una serie di funzioni e di blocchi funzione per la gestione della comunicazione multimaster su linea seriale. La possibilità di fare dialogare su una unica linea seriale RS422/485 più dispositivi master permette di velocizzare il dialogo tra i sistemi ottimizzando l'impegno della linea seriale.

Questa particolarità è molto apprezzata nelle connessioni radio, dove radiomodems che sfruttano tutti la stessa frequenza possono dialogare tra di loro minimizzando l'impegno della banda ed aumentando i tempi di trasferimento delle informazioni.

Nelle comunicazioni tra diversi sistemi su doppino seriale e/o con radiomodems tipicamente si utilizza un protocollo a pacchetto (Esempio modbus) ed uno dei sistemi che funge da master dialoga ciclicamente con tutti gli altri sistemi della rete scambiando le informazioni tra di loro. Come si può ben capire questa soluzione ha i seguenti difetti:

a) Tutta la comunicazione è delegata al sistema master, nel caso di guasto di quest'ultimo tutta la rete è ferma.
b) Il sistema master deve interrogare i vari sistemi slaves per conoscere se hanno dati da inviare al master od agli altri slaves. Questo implica un impegno del canale di comunicazione anche quando gli slaves non hanno informazioni utili da scambiare.
c) Lo scambio dati tra due sistemi slaves deve passare dal sistema master e questo rallenta l'invio dei dati da un sistema all'altro oltre ad un maggiore impegno del canale di comunicazione.

Utilizzando questa libreria è possibile fare dialogare i diversi sistemi direttamente tra di loro in modalità peer to peer. In questo modo un sistema invia dati all'altro solo quando è necessario garantendo un rapido invio dei dati con il minimo impegno del canale di comunicazione.

Comunicazione in broadcast

Nel caso in cui un dispositivo su di un nodo della rete sia in comunicazione con molti altri dispositivi ed i dati in scambio siano gli stessi per tutti i dispositivi è possibile utilizzare il blocco funzione ***BroadcastDataSend*** che invia il pacchetto dati in broadcast, tutti i dispositivi che hanno come nodo di destinazione ***ToNode*** nel blocco funzione ***DataTxferClient*** l'indirizzo del nodo che invia il messaggio in broacast lo riceveranno senza pero inviare l'acknowledge.

Questo significa che la trasmissione in broadcast permette di velocizzare il trasferimento dati da un nodo ad altri nodi permettendo altresì la ricezione simultanea dei dati inviati su tutti i nodi. Ma non è certa l'avvenuta ricezione dei dati da parte dei nodi destinatari, quindi conviene fare seguire ad una trasmissione in broadcast anche una trasmissione peer to peer. Nella figura seguente è riportato uno scenario tipico di comunicazione multimaster via radiomodem.

Il PLC nodo 0 scambia dati con i PLC nodo 1 e 2.

Il PLC nodo 1 scambia i dati con i PLC nodo 0, 2 e 3.

Il PLC nodo 2 scambia i dati con i PLC nodo 0, 1 e 3.

Il PLC nodo 3 scambia i dati con i PLC nodo 1 e 2.

Nel PLC nodo 0 è gestita anche la FB di invio in broadcast, attivando la trasmissione i dati saranno ricevuti solo dai PLC nodo 1 e 2 e non dal PLC nodo 3 perchè non ha attivo nessun scambio con il PLC del nodo che ha inviato il messaggio di broadcast.

PLC Node 2
MMasterDataTxfer
MyNode
2
DataTxferClient
ToNode
0
DataTxferClient
ToNode
1
DataTxferClient
ToNode
3
Radio Modem
Radio Modem
PLC Node 0
MMasterDataTxfer
MyNode
0
DataTxferClient
ToNode
1
DataTxferClient
ToNode
2
BroadcastDataSend
Radio Modem
PLC Node 1
MMasterDataTxfer
MyNode
1
DataTxferClient
ToNode
0
DataTxferClient
ToNode
2
DataTxferClient
ToNode
3
Radio Modem
PLC Node 3
MMasterDataTxfer
MyNode
3
DataTxferClient
ToNode
1
DataTxferClient
ToNode
2

Type	Library
FB	eMMasterDTxferLib_C000

7.22.1 MMasterDataTxfer, multimaster data transfer

MMasterDataTxfer
Enable — Enabled
File — Fault
MyNode — MMDtTxfID
MinFreeTm — RxFrames
MaxFreeTm — TxFrames
MinChkTm — RxErrors
MaxChkTm

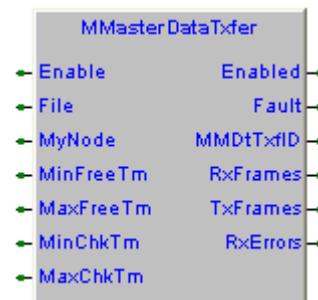
Questo blocco funzione esegue l'interfaccia con il terminale di I/O definito in ***File*** per gestire la comunicazione multimaster su rete multidrop. Questo è un blocco funzione protetto per utilizzarlo occorre richiedere il codice di protezione, vedi ***protezione funzioni e blocchi funzione***. E' comunque possibile utilizzarlo liberamente in modo test per 30 Min.

Il blocco funzione agisce da server di comunicazione gestendo il dispositivo di I/O, verranno poi connessi dei blocchi funzione client che gestiscono lo scambio dati tra i sistemi. L'FB ritorna un ***MMDtTxfID*** che deve essere passato alle FB client (Esempio ***DataTxferClient***).

In ***MyNode*** occorre definire il numero di nodo del sistema, in una rete il numero di nodo deve essere univoco. Tutti i messaggi che hanno come nodo di destinazione il valore di ***MyNode*** saranno ricevuti dalla FB che li passerà alle FB clients per la verifica.

La comunicazione multimaster si basa sul controllo del canale di comunicazione libero e sulla gestione delle collisioni, i parametri per questa gestione sono impostabili in ***MinFreeTm*** e ***MaxFreeTm***. In questi parametri occorrerà definire tempi piccoli per comunicazioni su linea seriale e tempi più lunghi nel caso di comunicazione via radiomodem.

In ***MinChkTm*** e ***MaxChkTm*** è possibile impostare un tempo per l'invio di un messaggio di controllo verso gli altri sistemi della rete. L'FB abiliterà a turno sequenzialmente le varie FB clients per uno scambio dati con il peer con cui l'FB dialoga controllando la connessione. ***Se uno o entrambi i valori sono impostati a 0 il controllo non viene eseguito***.

In ***RxErrors*** è ritornato il conteggio degli errori di ricezione, l'FB monitora continuamente il canale di comunicazione e se i dati ricevuti sono in errore viene incrementato il contatore. Errori possono manifestarsi in caso di collisioni sul canale di comunicazione. In caso di errore esecuzione viene attivata per un loop l'uscita ***Fault***.

Enable (BOOL) Comando di abilitazione blocco funzione.

File (FILEP) Flusso dati stream ritornato dalla funzione Sysfopen.

MyNode (USINT) Nodo identificativo del sistema (Range da 0 a 250).

Timeout (UDINT) Tempo attesa frame di acknowledge dal sistema peer (mS).

MinFreeTm (REAL) Tempo minimo attesa canale di comunicazione libero (S).

MaxFreeTm (REAL) Tempo massimo attesa canale di comunicazione libero (S).

MinChkTm (REAL) Tempo minimo attesa invio frame controllo verso sistema peer (S).

MaxChkTm (REAL) Tempo massimo attesa invio frame controllo verso sistema peer (S).

Enabled (BOOL) Attivo su abilitazione blocco funzione.

Fault (BOOL) Attivo per un loop su errore esecuzione.

MMDtTxfID (UDINT) ID server multimaster da passare alle FB clients (Esempio ***DataTxferClient***).

RxFrames (UDINT) Counter frame dati ricevuti. Sono conteggiati tutti i frames corretti indipendentemente se diretti a questo nodo.

TxFrames (UDINT) Counter frame dati trasmessi.

RxErrors (UDINT) Counter errori frame dati ricevuti.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10040010 Valore di ***File*** non definito.

10040020 FB protetta, terminato tempo funzionamento in modo demo.

10040050 Timeout esecuzione.

10040070 Errore case gestione.

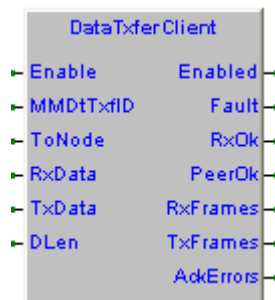
7.22.2 DataTxferClient, Data transfer client

Type	Library
FB	eMMasterDTxferLib_C000

Questo blocco funzione esegue lo scambio dati con un altro sistema su canale di comunicazione. Si collega al blocco funzione ***MMasterDataTxfer*** di gestione dispositivo di comunicazione, occorre passare ***MMDtTxfID*** in uscita dal blocco funzione server.

Il blocco funzione scambia i dati con il sistema definito in ***ToNode***. In pratica il valore di ***ToNode*** deve coincidere con il valore di ***MyNode*** della FB ***MmasterDataTxfer*** dell'altro sistema.

In ***RxData*** ed in ***TxData*** occorre definire l'indirizzo del buffer dati che si vuole scambiare con il sistema peer. In ***DLen*** la dimensione in bytes del buffer dati in scambio (I buffers ***RxData*** e ***TxData*** devono avere la stessa dimensione). L'FB controlla se vi è una variazione dei dati nel buffer ***TxData*** e ne esegue immediatamente l'invio al sistema peer che risponde con i dati del proprio buffer ***TxData*** che saranno trasferiti nel buffer ***RxData***.

L'uscita ***PeerOk*** è attiva se la comunicazione con il sistema peer è operativa, in caso di errori di comunicazione l'uscita si disattiva. In ***RxFrames*** e ***TxFrames*** è ritornato il conteggio dei frames dati ricevuti ed inviati dalla FB verso il sistema peer, in ***AckErrors*** il numero di errori di acknowledge da parte del sistema peer. In caso di errore esecuzione viene attivata per un loop l'uscita ***Fault***.

Enable (BOOL) Comando di abilitazione blocco funzione.

MMDtTxfID (UDINT) ID server multimaster in uscita dalla FB server (***MmasterDataTxfer***).

ToNode (USINT) Nodo identificativo del sistema peer con cui scambiare i dati (Range da 0 a 250).

RxData (@USINT) Puntatore al buffer dove devono essere trasferiti i dati ricevuti.

TxData (@USINT) Puntatore al buffer dove sono presenti i dati da trasmettere.

DLen (UDINT) Numero di bytes scambiati (Max 32).

Enabled (BOOL) Attivo su abilitazione blocco funzione.

Fault (BOOL) Attivo per un loop su errore esecuzione.

RxOk (BOOL) Attivo per un loop ad ogni ricezione dati da sistema peer.

PeerOk (BOOL) Attivo se scambio dati con sistema peer è Ok.

RxFrames (UDINT) Counter frame dati ricevuti.

TxFrames (UDINT) Counter frame dati trasmessi.

AckErrors (UDINT) Counter errori acknowledge dati da sistema peer.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10041010 ***MMDtTxfID*** non definito.

10041020 ***MMDtTxfID*** non corretto.

10041050 Valore **DLen** errato.

10041200 Frame dati ricevuto da sistema peer ha lunghezza errata. Verificare ***DLen*** sistema peer.

Esempi

Nell'esempio è gestito lo scambio di 8 BOOL con il sistema peer nodo 1 (***MyNode***=1).

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer
2	MMDtTxfID	UDINT	Auto	No	0	..	Multimaster data transfer ID
3	Client	DataTxferClient	Auto	No	0	..	Data transfer client
4	Server	MMasterDataTxfer	Auto	No	0	..	Multimaster data transfer
5	Dummy	BOOL	Auto	No	FALSE	..	Dummy variable
6	PeerOk	BOOL	Auto	No	FALSE	..	Peer system is Ok
7	RxData	BOOL	Auto	[0..7]	8(0)	..	Rx data from peer
8	TxData	BOOL	Auto	[0..7]	8(0)	..	Tx data to peer

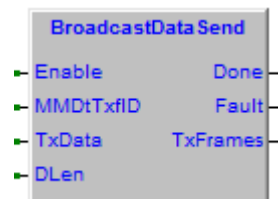
Esempio LD

SysFirstLoop
Sysfopen
EN ENO
'COM2' FName Fp
'rw' Mode
Server
MMasterDataTxfer
Enable Enabled Dummy (S)
Fp File Fault
0 MyNode MMDtTxfID MMDtTxfID
0.1 MinFreeTm RxFrames
0.2 MaxFreeTm TxFrames
10.0 MinChkTm RxErrors
20.0 MaxChkTm
Client
DataTxferClient
Enable Enabled Dummy
MMDtTxfID MMDtTxfID Fault PeerOk
1 ToNode RxOk
RxData ADR RxData PeerOk
TxData ADR TxData RxFrames
TxData SIZEOF DLen TxFrames
AckErrors

7.22.3 BroadcastDataSend, broadcast data send

Type	Library
FB	eMMasterDTxferLib_C000

Questo blocco funzione esegue l'invio di dati in modo broadcast. Si collega al blocco funzione ***MMasterDataTxfer*** di gestione dispositivo di comunicazione, occorre passare ***MMDtTxfID*** in uscita dal blocco funzione server.

Il blocco funzione invia i dati presenti nel buffer puntato da ***TxData*** con indirizzo di broadcast 16#FF. Tutti i sistemi che eseguono la FB ***DataTxferClient*** che hanno ***ToNode*** uguale al valore di ***Node*** definito nella ***MMasterDataTxfer*** riceveranno i dati inviati.

In ***TxData*** occorre definire l'indirizzo del buffer ed in ***DLen*** la dimensione in bytes del buffer dati che si vuole scambiare con i sistemi peer Ad ogni attivazione dell'ingresso ***Enable*** viene eseguito l'invio dei dati in broadcast, terminato l'invio si attiva l'uscita ***Done*** che rimane attiva sino alla disattivazione dell'ingresso ***Enable***. Per effettuare una nuova trasmissione occorre disattivare e poi riattivare l'ingresso ***Enable***.

Enable (BOOL)	Comando di abilitazione blocco funzione.
MMDtTxfID (UDINT)	ID server multimaster in uscita dalla FB server (***MmasterDataTxfer***).
TxData (@USINT)	Puntatore al buffer dove sono presenti i dati da trasmettere.
DLen (UDINT)	Numero di bytes scambiati (Max 32).
Done (BOOL)	Attivo al termine dell'invio dei dati, rimane attivo fino alla disabilitazione di ***Enable***.
Fault (BOOL)	Attivo per un loop su errore esecuzione.
TxFrames (UDINT)	Counter frame dati trasmessi.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

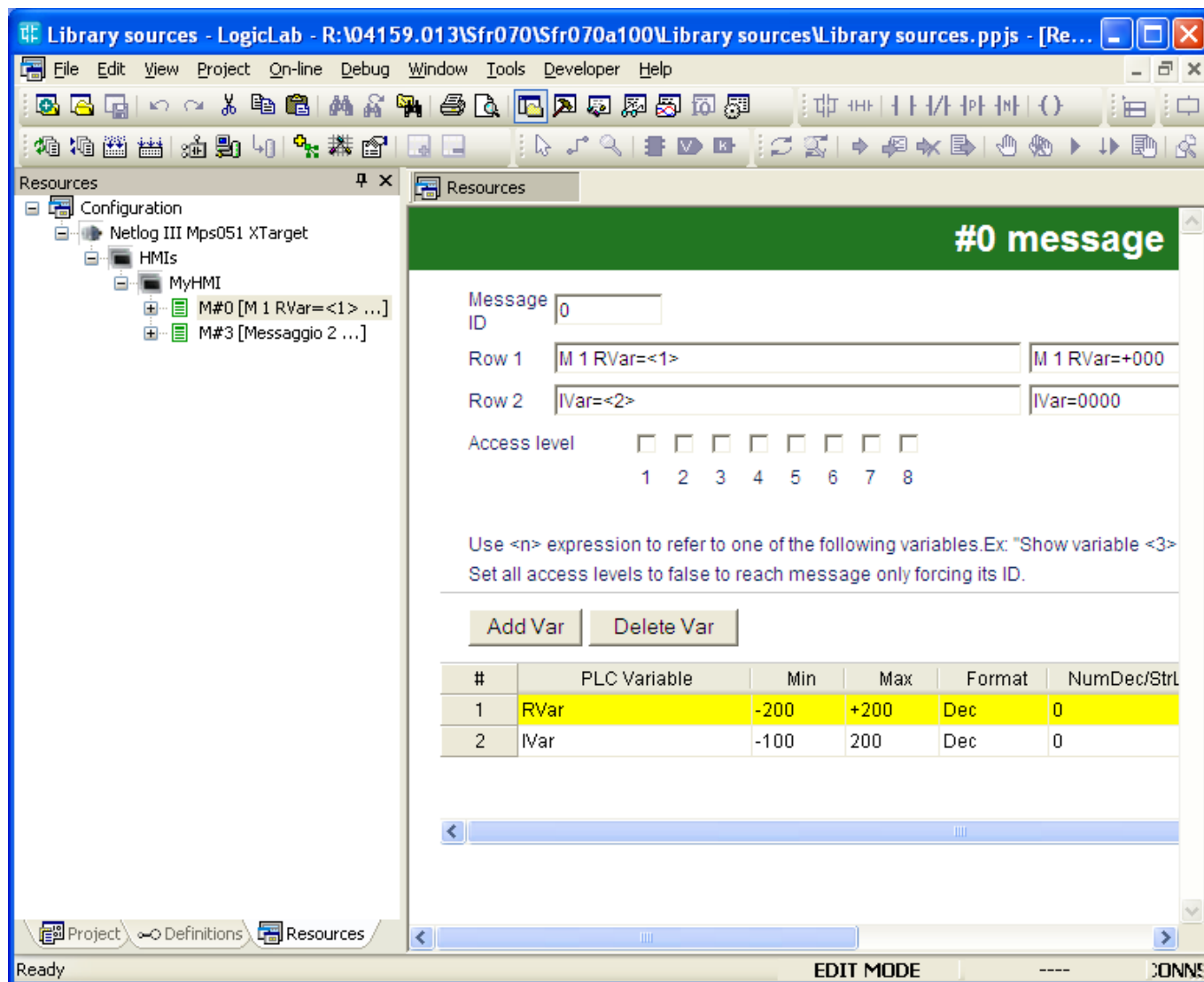
10044010 ***MMDtTxfID*** non definito.

10044020 ***MMDtTxfID*** non corretto.

10044050 Valore **DLen** errato.

7.23 Libreria gestione interfaccia operatore (eHMIBuiltInLib)

Attenzione! Per utilizzare la libreria occorre importarla nel proprio progetto. Vedere capitolo relativo all'import delle librerie.

Questa libreria permette di gestire l'interfaccia operatore direttamente da LogicLab. Nel menù Resources è possibile configurare le interfacce operatore. Come si vede nella figura è stata definita una interfaccia ***MyHMI*** che prevede due messaggi.

Come si vede ad ogni messaggio è associato del testo ed eventuali variabili da visualizzare. Nel messaggio possono essere visualizzate tutte le variabili globali.

Nella compilazione del progetto LogicLab crea ed istanzia per ogni HMI un blocco funzione di tipo ***HMIBuiltInMessages***, con il nome assegnato all'HMI, nell'esempio soprariportato ***MyHMI***. Questo FB provvede alla gestione dei messaggi e delle variabili ma per la visualizzazione occorre collegare al FB una FB di gestione terminale.

7.23.1 HMIBuiltInMessages, HMI built in messages

Type	Library
FB	eHMIBuiltInLib_B000

Questo blocco funzione esegue la gestione dei messaggi così come definiti da LogicLab.

Anche se ne riporto la raffigurazione a differenza degli altri blocchi funzione è automaticamente istanziato da LogicLab e non và quindi inserito dall'utente nel proprio programma. Per l'utente è importante fare riferimento alle sole variabili di ingresso ed uscita dal FB, variabili che possono essere utilizzate nel proprio programma.

Il parametro ***ALevel*** configura il livello di accesso ai messaggi, settando i bits della variabile si permette di visualizzare solo i messaggi che hanno lo stesso bit definito in ***Access Level***.

CmdDisable (BOOL) Disabilitazione comandi terminale. Attivandolo non è possibile eseguire operazioni.

ViewMID (UINT) Visualizza messaggio con ID definito.

ALevel (BYTE) Pattern livello accesso utente.

DEntryOk (BOOL) Attivo per un loop al termine della inputazione di una variabile.

MsgID (UINT) ID del messaggio attualmente visualizzato.

HMIBuildInID (UDINT) ID messaggi da passare alle FB collegate.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10046010 Tabella messaggi generata da LogicLab non supportata.

10046020~2 Non è possibile allocare memoria per l'esecuzione.

10046100 Blocco funzione non eseguito in task di background (Back).

10046200 Messaggio con ID richiesto in ***ViewMID*** non trovato.

10046100 Spazio insufficente per visualizzazione variabile.

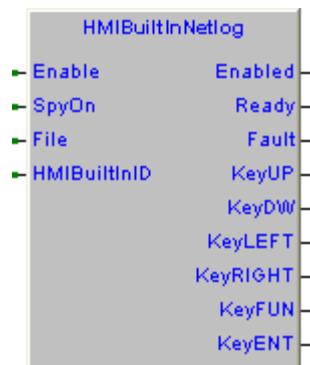
7.23.2 HMIBuiltInNetlog, Netlog HMI management

Type	Library
FB	eHMIBuiltInLib_B000

Questo blocco funzione esegue la gestione del terminale (Display e tastiera) integrato nel sistema ***NetlogIII***, ***deve essere eseguito in task Back***. Si collega al blocco funzione ***HMIBuiltInMessages*** di gestione messaggi, occorre passare ***HMIBuildInID*** in uscita dal blocco funzione.

Attivando l'ingresso ***Enable*** si attiva l'uscita ***Enabled*** e viene gestito il terminale visualizzando i messaggi definiti nel terminale da LogicLab. Nell'ingresso ***File*** occorre passare lo stream del dispositivo di I/O utilizzato per la gestione (Ritornato dalla funzione ***Sysfopen***).

Il FB ritorna in uscita lo stato di tutti i tasti del terminale, e questo permette di utilizzarli nel proprio programma come comandi. L'ingresso ***SpyOn*** se attivo permette di spiare il funzionamento della FB.

In caso di errore esecuzione viene attivata per un loop l'uscita ***Fault***.

Enable (BOOL) Comando di abilitazione blocco funzione.

SpyOn (BOOL) Se attivo permette di spiare il funzionamento della FB.

TFlags	Description
16#00000001	***Rx:*** Ricezione dati da terminale.
16#00000002	***Tx:*** Trasmissione comandi display.
16#00000004	***Tx:*** Trasmissione dati display.
16#00000008	***Tx:*** Trasmissione comando inputazione dati.

File (FILEP) Flusso dati ***stream*** ritornato dalla funzione ***Sysfopen***.

ALevel (USINT) Livello di accesso ai messaggi.

HMIBuiltInID (UDINT) ID gestione messaggi in uscita dalla FB (***HMIBuiltInMessages***).

Enabled (BOOL) Attivo su abilitazione blocco funzione.

Ready (BOOL) Attivo se l'hardware display è pronto.

Fault (BOOL) Attivo per un loop su errore esecuzione.

KeyUP (BOOL) Stato tasto UP del terminale.

KeyDW (BOOL) Stato tasto DW del terminale.

KeyLEFT (BOOL) Stato tasto LEFT del terminale.

KeyRIGHT (BOOL) Stato tasto RIGHT del terminale.

KeyFUN (BOOL) Stato tasto FUN del terminale.

KeyENT (BOOL) Stato tasto ENT del terminale.

Codici di errore

In caso di errore si attiva l'uscita ***Fault***, con ***SysGetLastError*** è possibile rilevare il codice di errore.

10047010 ***HMIBuildInID*** non definito.

10047020 ***HMIBuildInID*** non corretto.

10047100 Blocco funzione non eseguito in task di background (Back).

Esempi

Nell'esempio è gestito il terminale di un sistema ***NetlogIII***, i messaggi sono stati definiti nel programma LogicLab con il nome ***MyHMI***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	HMI	HMIBuiltInNetlog	Auto	No	0	..	FB Netlog terminal
2	MsgID	UINT	Auto	No	0	..	Displayed message ID

Esempio ST

```
(* Here open the terminal communication port. *)

IF (SysFirstLoop) THEN
    HMI.File:=Sysfopen('PCOM0.1', 'rw'); (* File pointer *)
END_IF;

(* Here manages the NETLOG III terminal. *)

MsgID:=MyHMI.MsgID; (* Displayed message ID *)
HMI.HMIBuiltInID:=MyHMI.HMIBuiltInID; (* ID gestione messaggi *)
HMI(Enable:=TRUE);
```

8 Protocolli di comunicazione

8.1 Protocollo modbus

Il modbus è un protocollo di comunicazione seriale diventato uno standard de facto nella comunicazione di tipo industriale, ed è ora il protocollo di connessione più diffuso fra i dispositivi elettronici industriali. E' un protocollo di tipo richiesta/risposta ed offre dei servizi specificati da function codes.

SlimLine supporta il protocollo modbus Rtu sulle porte seriali, e modbus Over IP su connessione ethernet su porta ***502***. Il protocollo modbus Rtu sulla porta seriale ha come parametri di comunicazione di default ***115200, e, 8, 1***, e l'indirizzo di nodo sia su porta seriale che su TCP/IP è ***1***.

8.1.1 Accesso variabili da modbus

Le funzioni del protocollo accedono tutte alla memoria utente ***MX100***, le funzioni supportate sono:

Code	Function	Tipo oggetto	Tipo accesso	Range indirizzo
01h	Read coil status	Bit singolo	Read	40000-44095 (20000-24095) ***(Note 1)***
02h	Read input status	Bit singolo	Read	40000-44095 (20000-24095) ***(Note 1)***
03h	Read holding registers	Word (16 Bit)	Read	40000-42047 (20000-22047) ***(Note 2)***
04h	Read input registers	Word (16 Bit)	Read	40000-42047 (20000-22047) ***(Note 2)***
05h	Force single coil	Bit singolo	Write	40000-44095 (20000-24095) ***(Note 1)***
06h	Preset single register	Word (16 Bit)	Write	40000-42047 (20000-22047) ***(Note 2)***
10h	Preset multiple registers	Word (16 Bit)	Write	40000-42047 (20000-22047) ***(Note 2)***

Da versione SFW167D000 l'area è indirizzabile anche nel range da 20000 a 2xxxx.

Note 1) Nelle funzioni che accedono al bit singolo (In realtà ogni bit equivale ad un byte di memoria) si utilizza nel comando l'indirizzo della variabile, quindi dovendo accedere alla locazione ***MX100.50*** utilizzeremo come indirizzo il valore ***40050***.

Note 2) Nelle funzioni che accedono ai registri (16 Bits) occorre considerare l'indirizzo della variabile diviso per 2, quindi dovendo raggiungere da modbus la locazione ***MX100.50*** utilizzeremo come indirizzo il valore ***40025***.

8.1.2 Lettura variabili da modbus

Per la lettura delle variabili si utilizza il comando ***Read holding registers*** (Codice ***0x03***). Ipotizzando di dover accedere in lettura ad una variabile ***DWORD*** allocata in memoria all'indirizzo ***MX100.64*** calcoleremo l'indirizzo di lettura nel modo:

((Indirizzo variabile/2)+Offset)-1 → ((64/2)+40000)-1=40031 → 0x9C5F

Essendo una variabile ***DWORD*** dovremo leggere 2 registri consecutivi a partire dal suo indirizzo di allocazione, ipotizzando che il valore della variabile sia ***0x12345678*** avremo.

Frames modbus RTU

Stringa di comando: ***01 03 9C 5F 00 02 DA 49***
Stringa di risposta: ***01 03 04 56 78 12 34 66 D5***

Frames modbus TCP/IP

Stringa di comando: ***00 00 00 00 00 06 01 03 9C 5F 00 02***
Stringa di risposta: ***00 00 00 00 00 07 01 03 04 56 78 12 34***

La rappresentazione dei dati in SlimLine è nel formato ***Little-Endian***, la numerazione inizia dal byte meno significativo per finire col più significativo, quindi come si nota dalla stringa di risposta il valore della variabile a 32 bits ***0x12345678*** viene ritornato suddiviso in due registri a 16 bits con i valori ***0x5678***, ***0x1234***.

8.1.3 Scrittura variabili da modbus

Per la scrittura delle variabili si utilizza il comando ***Preset multiple registers*** (Codice ***0x10***). Ipotizzando di dover accedere in scrittura ad una variabile ***DWORD*** allocata in memoria all'indirizzo ***MX100.64*** calcoleremo l'indirizzo di scrittura nel modo:

((Indirizzo variabile/2)+Offset)-1 → ((64/2)+40000)-1=40031 → 0x9C5F

Essendo una variabile ***DWORD*** dovremo scrivere 2 registri consecutivi a partire dal suo indirizzo di allocazione, ipotizzando di dover scrivere nella variabile il valore ***0x12345678*** avremo.

Frames modbus RTU

Stringa di comando: ***01 10 9C 5F 00 02 04 56 78 12 34 D3 33***
Stringa di risposta: ***01 10 9C 5F 00 02 5F 8A***

Frames modbus TCP/IP

Stringa di comando: ***00 00 00 00 00 0B 01 10 9C 5F 00 02 04 56 78 12 34***
Stringa di risposta: ***00 00 00 00 00 06 01 10 9C 5F 00 02***

La rappresentazione dei dati in SlimLine è nel formato ***Little-Endian***, la numerazione inizia dal byte meno significativo per finire col più significativo, quindi come si nota dalla stringa di comando il valore da scrivere a 32 bits ***0x12345678*** viene definito suddiviso in due registri a 16 bits con i valori ***0x5678***, ***0x1234***.

8.1.4 Accesso Real time clock da modbus

E' possibile accedere al real time clock utilizzando i comandi modbus di accesso ai registri le funzioni supportate sono:

Code	Function	Tipo oggetto	Tipo accesso	Range indirizzo
03h	Read holding registers	Word (16 Bit)	Read	100-105 (150 per l'Epoch time)
04h	Read input registers	Word (16 Bit)	Read	100-105 (150 per l'Epoch time)
06h	Preset single register	Word (16 Bit)	Write	100-105 (150 per l'Epoch time)
10h	Preset multiple registers	Word (16 Bit)	Write	100-105 (150 per l'Epoch time)

I registri (16 Bits) del real time clock sono allocati in locazioni consecutive a partire dall'indirizzo modbus 100. I registri contengono il valore attuale del real time clock e scrivendo un nuovo valore il real time clock verrà automaticamente aggiornato.

Address	Register	Note
100	Second	Valore secondi (Range da 0 a 59)
101	Minute	Valore minuti (Range da 0 a 59)
102	Hour	Valore ora (Range da 0 a 23)
103	Day	Valore giorno (Range da 1 a 31)
104	Month	Valore mese (Range da 1 a 12)
105	Year	Valore anno (Range da 1900 a 2037)

8.1.5 Lettura RTC da modbus

Per la lettura dei registri del real time clock si utilizza il comando ***Read holding registers*** (Codice ***0x03***). Dovremo leggere 6 registri consecutivi a partire dall'indirizzo di allocazione, l'indirizzamento di modbus prevede un offset di 1, quindi ***99*** (***0x0063***).

Frames modbus RTU

Stringa di comando: ***01 03 00 63 00 06 35 D6***
Stringa di risposta: ***01 03 0C 00 1E 00 30 00 0B 00 1D 00 09 07 DA A2 32***

Frames modbus TCP/IP

Stringa di comando: ***00 00 00 00 00 06 01 03 00 63 00 06***
Stringa di risposta: ***00 00 00 00 00 0F 01 03 0C 00 1E 00 30 00 0B 00 1D 00 09 07 DA***

Come si vede dalla risposta il valore è:

Secondi: 30 (***0x001E***)
Minuti: 48 (***0x0030***)
Ora: 11 (***0x000B***)
Giorno: 29 (***0x001D***)
Mese: 9 (***0x0009***)
Anno: 2010 (**0x07DA**)

8.1.6 Scrittura RTC da modbus

Per la scrittura dei registri del real time clock si utilizza il comando ***Preset multiple registers*** (Codice ***0x10***). Dovremo scrivere 6 registri consecutivi a partire dall'indirizzo di allocazione, l'indirizzamento di modbus prevede un offset di 1, quindi ***99*** (***0x0063***). Ipotizziamo di dover impostare nel real time clock i valori:

Secondi: 30 (***0x001E***)
Minuti: 48 (***0x0030***)
Ora: 11 (***0x000B***)
Giorno: 29 (***0x001D***)
Mese: 9 (***0x0009***)
Anno: 2010 (**0x07DA**)

Frames modbus RTU

Stringa di comando: ***01 10 00 63 00 06 08 00 1E 00 30 00 0B 00 1D 00 09 07 DA 5D C8***
Stringa di risposta: ***01 10 00 63 00 06 B0 15***

Frames modbus TCP/IP

Stringa di comando: ***00 00 00 00 00 13 01 10 00 63 00 06 08 00 1E 00 30 00 0B 00 1D 00 09 07 DA***
Stringa di risposta: ***00 00 00 00 00 06 01 10 00 63 00 06***

8.1.7 Accesso Epoch time da modbus

E' allocato anche un registro a 32 bits per il valore di data/ora in Epoch time, l'acceso a questo registro in lettura e/o scrittura và sempre effettuato usando due registri a 16 bits.

Address	Register	Note
150	Epoch time	Epoch time

8.1.8 Lettura Epoch time da modbus

Per la lettura dell'epoch time si utilizza il comando ***Read holding registers*** (Codice ***0x03***). Dovremo leggere 2 registri consecutivi a partire dall'indirizzo di allocazione, l'indirizzamento di modbus prevede un offset di 1, quindi ***149*** (***0x0095***).

Frames modbus RTU

Stringa di comando: ***01 03 00 95 00 02 D4 27***
Stringa di risposta: ***01 03 04 30 B5 4C A3 90 6C***

Frames modbus TCP/IP

Stringa di comando: ***00 00 00 00 00 06 01 03 00 95 00 02***
Stringa di risposta: ***00 00 00 00 00 07 01 03 04 30 B5 4C A3***

Come si vede dalla risposta il valore è: ***0x4CA330B5 → 1285763253 → GMT: Wed, 29 Sep 2010 12:27:33 UTC***.

8.1.9 Scrittura Epoch time da modbus

Per la scrittura dell'epoch time si utilizza il comando ***Preset multiple registers*** (Codice ***0x10***). Dovremo scrivere 2 registri consecutivi a partire dall'indirizzo di allocazione, l'indirizzamento di modbus prevede un offset di 1, quindi ***149*** (***0x0095***). Ipotizziamo di dover impostare il valore:

GMT: Wed, 29 Sep 2010 12:27:33 UTC → 1285763253 → 0x4CA330B5

Frames modbus RTU

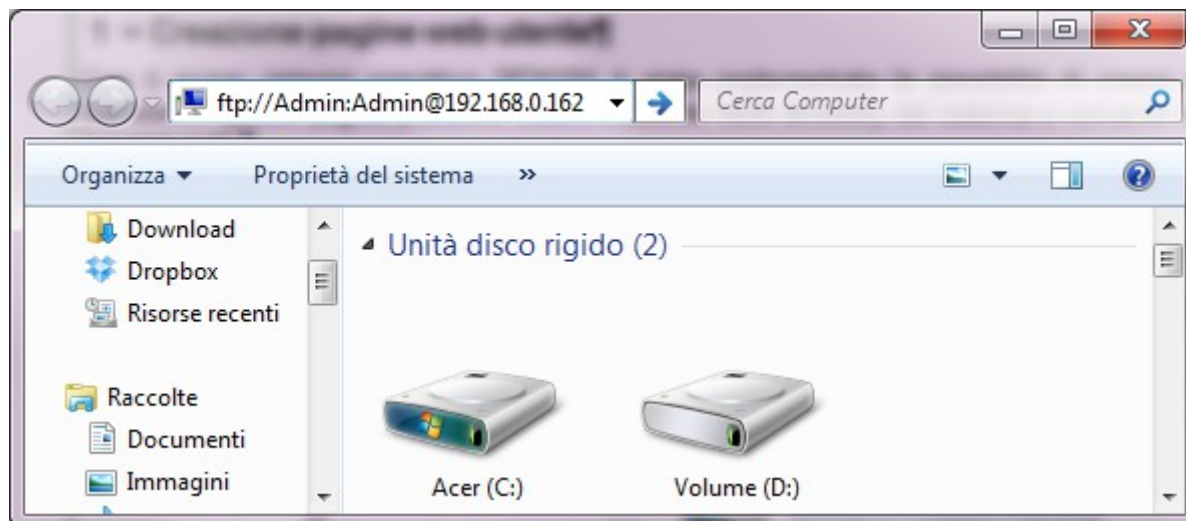
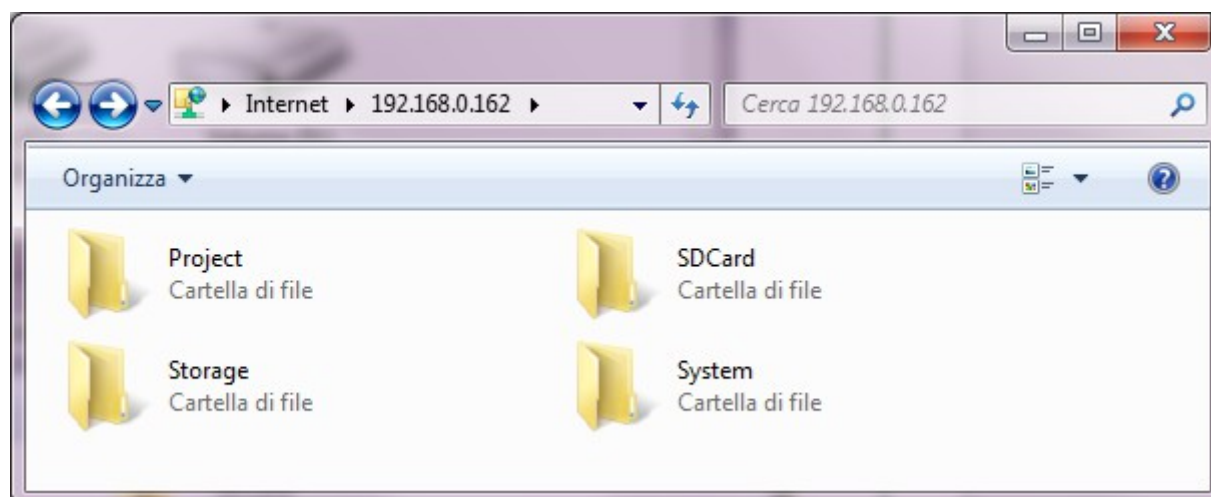
Stringa di comando: ***01 10 00 95 00 02 04 30 B5 4C A3 50 A3***
Stringa di risposta: ***01 10 00 95 00 02 51 E4***

Frames modbus TCP/IP

Stringa di comando: ***00 00 00 00 00 0B 01 10 00 95 00 02 04 30 B5 4C A3***
Stringa di risposta: ***00 00 00 00 00 06 01 10 00 95 00 02***

9 Creazione pagine web utente

Con il nuovo sistema operativo SFW184 è stata implementata la possibilità di creare pagine web direttamente dall'utente, queste pagine potranno essere trasferite nelle directory del sistema e verranno visualizzate accedendo da browser web.

Per trasferire le pagine web create dall'utente nel file system del sistema SlimLine occorre utilizzare un client Ftp (Esempio FileZillla) ma è possibile usare anche il semplice esplora risorse di Windows. Come si vede dalla figura sottostante, impostando nella barra indirizzo le credenziali di accesso al sistema ed il suo indirizzo IP ***ftp://Admin:Admin@192.168.0.162***, è possibile connettersi e visualizzare il file system.

Ecco come si presenta la visualizzazione del file system alla connessione. Le cartelle ***Project*** e ***System*** sono riservate al sistema e si consiglia di ***non modificarne il contenuto***. I file delle pagine utente possono essere trasferiti nelle cartelle ***Storage*** e ***SDCard*** (Se presente).

Quindi l'utente può creare le sue pagine web utilizzando un qualsiasi editor html ma anche semplicemente usando un semplice editor di testo come il blocco note, certo deve conoscere la sintassi del linguaggio html. Le pagine create saranno trasferite nella directory desiderata ed accedendo da un normale browser alla pagina la pagina sarà visualizzata.

9.1 Criteri per realizzazione pagina

Naturalmente il web server integrato nello SlimLine ha solo un ridotto set di funzioni e quindi nella creazione delle pagine web occorre sottostare a certe regole, vediamole:

a) La pagina non può contenere inclusione di altre pagine (Esempio pagine di stile o di scripts).

b) La pagina non può contenere inclusione di immagini (Esempio file gif o jpg), eventuali immagini possono essere embedded nella pagina stessa.

Vediamo ad esempio una semplice pagina che visualizza un messaggio di presentazione.

Sorgente html pagina

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>SlimLine - Simple page</title>
</head>
<body>
This page is served by the <b>SlimLine</b>
</body>
</html>
```

Salvando il testo riportato in un file, esempio ***SPage.htm***, e trasferendo il file nella directory ***Storage*** di SlimLine, sarà possibile visualizzare la pagina web risultante semplicemente digitando nel proprio browser l'indirizzo della pagina.

Naturalmente la pagina può contenere links ad altre pagine, sarà così possibile realizzare una propria navigazione tra diverse pagine. Ecco lo stesso esempio di prima con incluso la definizione di uno stile.

Sorgente html pagina

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>SlimLine - Simple page</title>
<style type="text/css">
.Bolded {font-family: Arial, Helvetica, sans-serif; font-size: 20px; font-style: normal;font-weight: bold;}
</style>
</head>
<body>
This page is served by the <span class="Bolded">SlimLine</span>
</body>
</html>
```

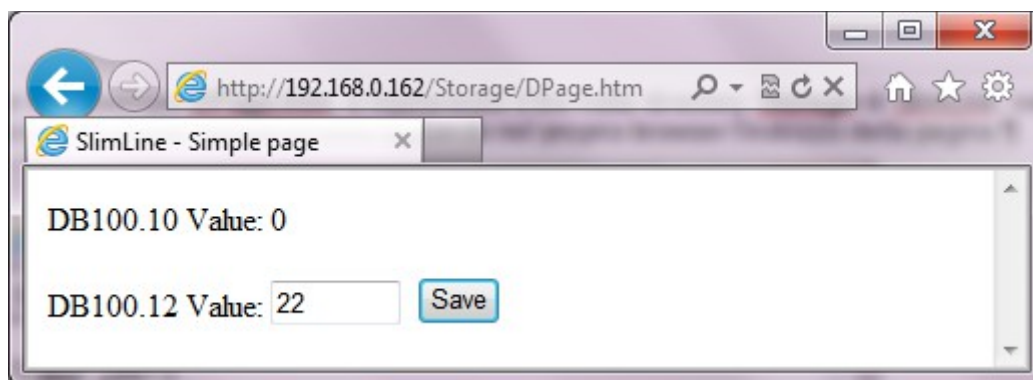
9.2 Pagine dinamiche

La caratteristica più importante del server web integrato in SlimLine è possibilità di gestire le ***pagine dinamiche***. Una pagina web dinamica è una pagina il cui contenuto, in tutto o in parte, è generato sul momento dal server, potendo dunque essere diversa ogni volta che viene richiamata, consentendo quindi un'interattività con l'utente.

Ecco quindi che sarà possibile realizzare pagine che riportano valori di variabili PLC e permettono di modificare il valore di variabili PLC. Nell'esempio seguente riporto il sorgente html di una semplice pagine che visualizza il valore di una variabile PLC di tipo UINT allocata all'indirizzo DB100.10 e permette di impostare il valore di una variabile PLC di tipo UINT allocata all'indirizzo DB100.12.

Sorgente html pagina

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>SlimLine - Simple page</title>
</head>
<body">
DB100.10: <!--["%d", UINT, 10]--></br>
<form id="MyForm" name="MyForm" method="post" action="DPage.htm">
DB100.12: <input name="UINT 12" type="text" size="5" maxlength="10" value="<!--["%d", UINT, 12]-->">;
 <input type="submit" id="MyButton" value="Save"/>
</form>
</body>
</html>
```

Salvando il testo riportato in un file, esempio ***DPage.htm***, e trasferendo il file nella directory ***Storage*** di SlimLine, sarà possibile visualizzare la pagina web risultante semplicemente digitando nel proprio browser l'indirizzo della pagina.

Come si vede nella riga superiore viene visualizzato il valore della variabile PLC DB100.10 mentre impostando un valore nella casella di testo della riga inferiore e agendo sul tasto ***Save*** sarà possibile impostare il valore della variabile PLC DB100.12.

Naturalmente in una pagina web possono essere visualizzate e possono essere impostate tutte le variabili desiderate, si consiglia comunque di non esagerare con il numero di variabili, è preferibile suddividerle in più pagine.

9.3 Formato TAGs

Come si è visto in una pagina dinamica parte del contenuto viene generato sul momento dal server Http (Il modulo CPU SlimLine), vediamo quali sono i meccanismi per definire le ***TAGs*** da visualizzare. All'interno del sorgente della pagina Html è possibile definire dei campi di commento del tipo <!--["%d", UINT, 10]-->.

I campi sono interpretati come commenti e quindi sono gestibili da qualsiasi editor Html (Esempio Macromedia), ma il server Http nel momento in cui invia la pagina al client (Il browser che la visualizza) sostituisce al campo il valore della variabile indicata. Nell TAG sono riportate tutte le informazioni necessarie secondo la sintassi

<!--[Format, Type, Address]-->

9.3.1 Campo Format

La stringa di formattazione ***Format***, può contenere elementi di due tipi, il primo consiste in caratteri che vengono ritornati nella pagina inalterati. Il secondo consiste in direttive di conversione che descrivono il modo in cui gli argomenti devono essere visualizzati. Le direttive di conversione iniziano con il simbolo % seguito dalle direttive secondo il formato:

% [Flags] [Width] [.Precision] [Length] Conversion

Flags

+	La visualizzazione delle variabili con segno, inizierà sempre con il segno - o +.
space	La visualizzazione delle variabili con segno, inizierà sempre con il segno - o con lo spazio.
x	I valori diversi da 0 vengono prefissati con 0x.
0	Al valore visualizzato vengono aggiunti 0 fino al raggiungimento del numero di cifre desiderato (Per variabili di tipo d, i, o, u, x, X, e, E, f, g, G).

Width: Definisce il numero di cifre che devono essere visualizzate.

Precision: Definisce il numero di cifre decimali da visualizzare (Per variabili di tipo e, E, f).

Length

h	Prima di (d, i, u, x, X, o) denota una variabile short int o unsigned short int.
l (elle)	Prima di (d, i, u, x, X, o) denota una variabile long int o unsigned long int.
L	Prima di (e, E, f, g, G) denota una variabile long double.

Conversion

d	Valore decimale con segno.
i	Valore decimale con segno.
o	Valore ottale senza segno.
u	Valore decimale senza segno.
x	Valore esadecimale, viene visualizzato utilizzando lettere minuscole (Da 0 a 9, da a a f).
X	Valore esadecimale, viene visualizzato utilizzando lettere maiuscole (Da 0 a 9, da A a F).
e	Valore decimale in virgola mobile, visualizzato con indicazione dell'esponente (Esempio: [-]d.ddde+dd).
E	Valore decimale in virgola mobile, visualizzato con indicazione dell'esponente (Esempio: [-]d.dddE+dd).
f	Valore decimale in virgola mobile (Esempio: [-]d.ddd).
c	Singolo carattere.
s	Stringa.

9.3.2 Campo Type

Il campo ***Type*** indica il tipo di variabile che si vuole visualizzare, sono gestiti tutti i tipi definiti nella IEC61131.

9.3.3 Campo Address

Il campo Address indica l'indirizzo della variabile, ricordo che è possibile indicare solo variabili allocate nella DB 100.

9.3.4 Esempi di TAGs

Per meglio comprendere il formato di visualizzazione delle TAGs riporto alcuni esempi.

<!--["%d", UINT, 10]-->	Visualizza il valore della variabile UINT allocata all'indirizzo DB 100.10 con un numero di cifre intere variabili in base al valore.
<!--["%04d", UINT, 10]-->	Visualizza il valore della variabile UINT allocata all'indirizzo DB 100.10 sempre espresso con 4 cifre.
<!--["%3.0f", REAL, 32]-->	Visualizza il valore della variabile REAL allocata all'indirizzo DB 100.32 con 3 cifre intere e nessuna cifra decimale.
<!--["%4.2f", REAL, 50]-->	Visualizza il valore della variabile REAL allocata all'indirizzo DB 100.50 con 2 cifre intere e 2 cifre decimali.

9.4 Formato ARGs

La principale peculiarità del web dinamico è la possibilità di variare i contenuti delle pagine in base alle richieste degli utenti. Questa possibilità si materializza attraverso i meccanismi che permettono agli utenti, oltre che di richiedere una pagina ad un web server, anche di specificare determinati parametri da inviare al server web. Per impostare da pagina web valori di variabili PLC viene gestita una richiesta di tipo POST, il metodo è utilizzato con i moduli: quando una pagina Html contiene un tag ***<form>***. I dati impostati nei vari oggetti contenuti nel ***<form>*** sono inviati in maniera da non essere direttamente visibili per l'utente, attraverso la richiesta HTTP che il browser invia al server.

Se ci riferiamo all'esempio precedente vediamo che la parte di pagina Html che permette l'impostazione della variabile PLC UINT allocata all'indirizzo DB 100.12 è la seguente.

Sorgente html pagina

```
<form id="MyForm" name="MyForm" method="post" action="DPage.htm">
DB100.12: <input name="UINT 12" type="text" size="5" maxlength="10" value="<!--["%d", UINT, 12]-->”>
 <input type="submit" id="MyButton" value="Save"/>
</form>
```

In pratica un campo ***<form>*** con id ***MyForm*** contiene una casella di testo con id ***UINT12*** di dimensione 5 caratteri con un massimo impostabile di 10 caratteri. Nel form trova posto anche un pulsante di tipo submit la cui pressione esegue l'invio dell'intero modulo.

Definendo nel browser il valore della casella di testo ed agendo sul tasto ***Save***, il dato inputato verrà inviato al server che visualizzerà la pagina ***DPage.htm*** e contemporaneamente provvederà a scrivere il valore definito nella variabile UINT DB100.12.

9.4.1 ARG name

Il campo ***name*** dell'argomento è molto importante in quanto definisce il tipo di variabile PLC da impostare (Sono gestiti tutti i tipi definiti nella IEC61131) ed il suo indirizzo, i due campi devono essere separati da uno spazio.

Un nome del tipo ***UINT 12*** indicherà una variabile UINT allocata ad indirizzo DB 100.12.
Un nome del tipo ***REAL 128*** indicherà una variabile REAL allocata ad indirizzo DB 100.128.
Quindi un nome del tipo ***STRING 1000 16*** indicherà una variabile STRING di lunghezza 16 caratteri allocata ad indirizzo DB 100.1000.

9.4.2 ARG id

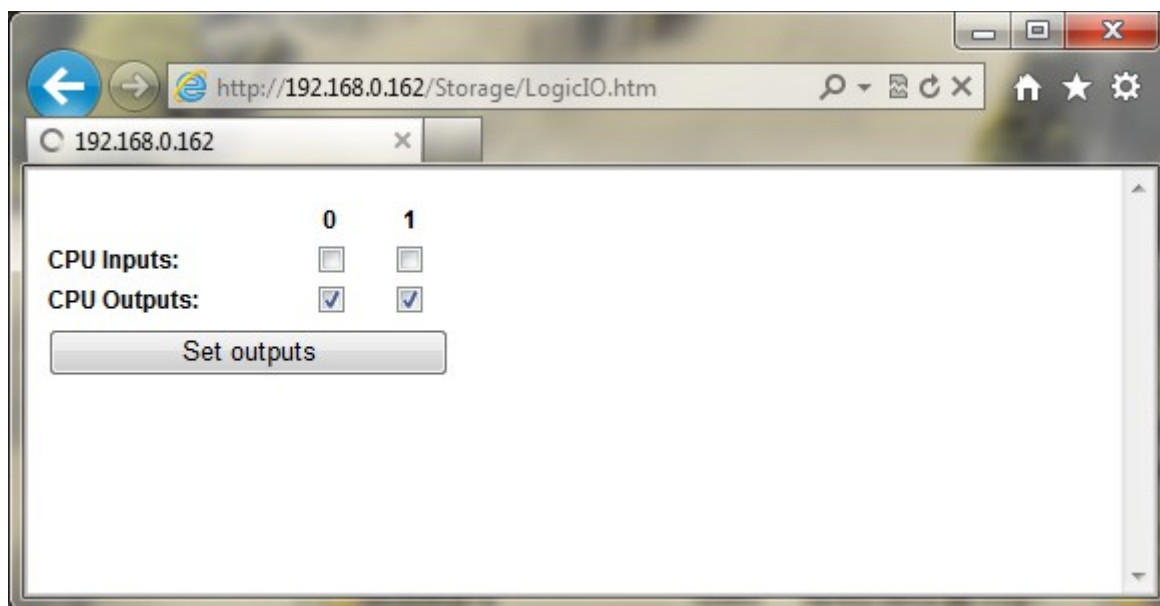
Il campo ***id*** dell'argomento serve a referenziare l'oggetto all'interno del form in modo da poterlo valorizzare con la funzione **SetValues()**. La scelta di definirlo come UINT12 utilizzata nell'esempio è puramente indicativa, meglio sarebbe utilizzare una definizione che ne riprenda il significato (Esempio “SetPoint”, “Preset”, ecc.).

9.5 Alcuni esempi

Naturalmente le pagine web vanno create in base alle proprie esigenze inserendo gli oggetti desiderati. Per facilitare lo sviluppo delle proprie pagine viene fornito un programma dimostrativo PTP128*000 che contiene una serie di programmi SlimLine e relative pagine web.

Per testare i vari programmi occorre trasferire sul modulo CPU il programma tramite LogicLab e tramite un client FTP trasferire la pagina ***htm*** nella directory ***Storage***. Ora da un browser si digita l'indirizzo IP del modulo CPU seguito dalla directory e dal nome della pagina Esempio ***http://192.168.0.122/Storage/Page.htm***.

9.6 LogicIO, gestione I/O logici

Ecco un esempio di gestione I/O logici da pagina web, per visualizzare lo stato degli ingressi e delle uscite sono stati utilizzati degli oggetti ***checkbox***. Lo stato di attivo è indicato dalla presenza del tick, per attivare le uscite si pone il tick sulla uscita desiderata e si agisce sul tasto ***Set outputs***.

Per visualizzare lo stato reale degli ingressi la pagina viene automaticamente rinfrescata ogni 10 secondi. Per ottenere l'aggiornamento della pagina dopo la direttiva ***<head>*** viene posta la dichiarazione:

<meta http-equiv="refresh" content="10">

Per la gestione della pagina sono utilizzate alcune funzioni javascript.

Check(Field, Value), Imposta o rimuove il simbolo di tick sull'oggetto di tipo checkbox indicato in ***Field*** in base a ***Value***.

SetValues(), Eseguita al caricamento della pagina esegue l'aggiornamento di tutti gli oggetti di tipo checkbox presenti.

SubmitForm(Form), Eseguita su pressione del tasto Set outputs controlla se i checkbox di attivazione uscite sono settati ed aggiorna il valore dei campi hidden di scrittura variabili.

Sorgente funzioni javascript di pagina

```
<script language="javascript">
function Check(Field, Value) {document.MyForm[Field].checked=(Value != 0);}
function SetValues()
{
    Check("Inp00", '<!--["%d", BOOL, 0]-->');
    Check("Inp01", '<!--["%d", BOOL, 1]-->');
    Check("Out00", '<!--["%d", BOOL, 3]-->');
    Check("Out01", '<!--["%d", BOOL, 4]-->');
}

function SubmitForm(Form)
{
    if (document.getElementById('Out00').checked) document.getElementById('BOOL3').value="1";
    if (document.getElementById('Out01').checked) document.getElementById('BOOL4').value="1";
    document.forms[Form].submit();
}
</script>
```

9.7 COMPort, parametri comunicazione seriale

Ecco un esempio di come gestire la visualizzazione e l'impostazione dei parametri di comunicazione seriale da pagina web. Per la selezione della porta, baud rate, numero bit dato, tipo di parità e modo di gestione del segnale DTR si sono utilizzati oggetti ***select***. L'oggetto select permette di selezionare il valore desiderato tramite una lista di valori.

Per le definizione dei tempi di ritardo sul segnale DTR si sono utilizzati degli oggetti ***text*** che permettono di visualizzare ed impostare valori.

Richiamando la pagina vengono visualizzati i valori corrispondenti alle impostazioni definite nelle variabili dello SlimLine. I valori sono stati appoggiati su di un'area ritentiva per mantenerne il valore impostato anche allo spegnimento del sistema.

Definendo nuovi valori ed agendo sul tasto ***Preset COM port*** i valori saranno trasferiti nelle variabili dello SlimLine e la pagina verrà visualizzata con i nuovi valori definiti.

Ecco come si presenta la pagina web, è stata visualizzata sulla finestra del programma LogicLab che visualizza in debug i valori delle variabili.

9.7.1 Funzioni javascript

Per la gestione della pagina sono utilizzate alcune funzioni javascript.

Set(Field, Value), Imposta il campo ***text*** di name ***Field*** con il valore ***Value***.

Choose(Field, Value), Seleziona l'oggetto ***select*** di name ***Field*** sull'opzione il cui valore corrisponde a ***Value***.

SetValues(), Eseguita al caricamento della pagina esegue l'aggiornamento di tutti gli oggetti presenti.

Sorgente funzioni javascript di pagina

```
<script language="javascript">
function Set(Field, Value) {document.MyForm[Field].value=Value;}
function Choose(Field, Value)
{
    for (i=0; i<document.MyForm[Field].options.length; i++)
        if (document.MyForm[Field].options[i].value == Value) document.MyForm[Field].selectedIndex=i;
}

function SetValues()
{
    Choose("Port", '<!--["%s", STRING, 2048]-->');
    Choose("Baudrate", '<!--["%d", UDINT, 2056]-->');
    Choose("DataBits", '<!--["%d", USINT, 2060]-->');
    Choose("Parity", '<!--["%d", STRING, 2064]-->');
    Choose("DTRMng", '<!--["%d", USINT, 2068]-->');
    Set("DTROn", '<!--["%d", UINT, 2070]-->');
    Set("DTROff", '<!--["%d", UINT, 2072]-->');}
</script>
```

9.8 Aggiornamento pagine con AJAX

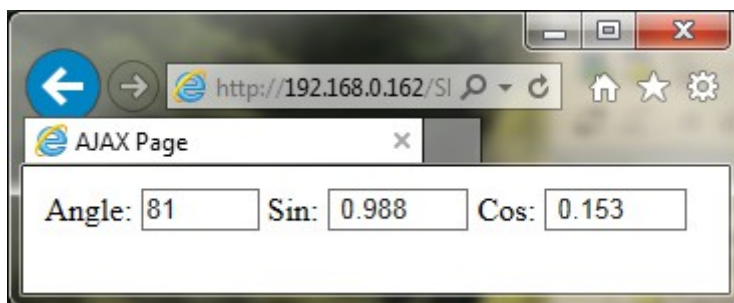
AJAX, acronimo di Asynchronous JavaScript and XML, è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive. Lo sviluppo di applicazioni HTML con AJAX si basa su uno scambio di dati in background fra web browser e server, che consente l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente.

Questa tecnica permette di avere un aggiornamento automatico dei dati in una pagina web senza dover ricaricare la pagina, permettendo di visualizzare le variabili PLC automaticamente. Vediamo come funziona questa tecnica, nella pagina web deve essere inserito uno script java che gestisce le richieste AJAX. Viene fornito uno script già pronto all'uso (Ns codice SFW191*000).

Ecco come si presenta una pagina web che visualizza il valore di un angolo ed i rispettivi valori di seno e coseno. I valori sono gestiti dallo SlimLine che ne esegue la variazione a tempo. I valori sono appoggiati in 3 variabili nella DB 100.

Angle UINT DB 100.0 Contiene valore angolo

Sin REAL DB 100.4 Contiene valore del seno

Cos REAL DB 100.8 Contiene valore del coseno

Sorgente html pagina

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>AJAX Page</title>
<script type="text/javascript">

// ********************************************************************************
// "SFW191A000" FUNZIONI PER GESTIONE AJAX
// ********************************************************************************
// Le seguenti funzioni gestiscono lo standard AJAX "Asynchronous Java and XML",
// con esse viene gestito lo scambio dinamico di dati con le pagine web.

var XMLHttp=AJAXCreateReqObject();function AJAXCreateReqObject(){var b=null;var
a=navigator.userAgent.toUpperCase();if(window.XMLHttpRequest){b=new
XMLHttpRequest()}else{if(window.ActiveXObject&&(a.indexOf("MSIE 4")<0)){if(a.indexOf("MSIE 5")<0){b=new
ActiveXObject("Msxml2.XMLHTTP")}else{b=new ActiveXObject("Microsoft.XMLHTTP")}}}return(b)}function
AJAXSendRequest(b){var a=Math.random();if(XMLHttp!=null){XMLHttp.open("GET",b+"?
Rnd="+escape(a),true);XMLHttp.setRequestHeader("connection","close");XMLHttp.onreadystatechange=AJAXHandleR
sp;XMLHttp.send(null)}}function AJAXHandleRsp(){switch(XMLHttp.readyState){case 0:break;case 1:break;case
2:break;case 3:break;case 4:if(XMLHttp.status==200){SetupValues(XMLHttp.responseText)}break}};

// ********************************************************************************
// FUNZIONE "SetupValues(PContent)"
// ********************************************************************************
// Questa funzione viene eseguita su risposta Ajax, nella variabile "PContent"
// è presente tutto il contenuto della pagina richiesta.
// ---------------------------------------------------------------------------

function SetupValues(PContent)
{
   var Value=new Array(); //Array valori ricevuti da server

   // Eseguo separazione valori, sono separati dal simbolo "|".

   if (PContent.indexOf('|') != -1)
   {
```

```
        Value=PContent.split('|');
        document.getElementById("Angle").value=Value[0];
        document.getElementById("Sin").value=Value[1];
        document.getElementById("Cos").value=Value[2];
    }
}

</script>
</head>
<body onLoad="setInterval('AJAXSendRequest(\'Values.htm\')', 3000)">
<table border="0">
  <tr>
    <td>Angle:</td>
    <td><input type="text" id="Angle" size="4" maxlength="4"/></td>
    <td>Sin:</td>
    <td><input type="text" id="Sin" size="6" maxlength="6"/></td>
    <td>Cos:</td>
    <td><input type="text" id="Cos" size="6" maxlength="6"/></td>
  </tr>
</table>
</body>
</html>
```

Sul caricamento della pagina ***<body onLoad="setInterval('AJAXSendRequest(\'Values.htm\')', 3000)">*** viene eseguita la richiesta AJAX ogni 3 secondi della pagina ***Values.htm***. Il valore di ritorno di questa pagina viene automaticamente passato alla funzione ***SetupValues*** che lo parsa e copia i valori negli oggetti di visualizzazione. La pagina ***Values.htm*** ritorna i valori delle 3 variabili separati dal simbolo **|**. Ecco il listato di questa pagina.

Sorgente pagina Values.htm

```
<!--['%d', UINT, 0]-->|<!--['%6.3f', REAL, 4]-->|<!--['%6.3f', REAL, 8]-->
```

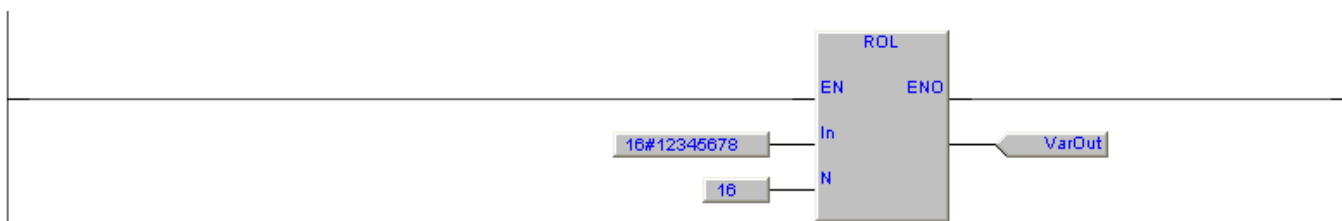
10 Tips and tricks

10.1 Swap variabile DWORD

Ecco come utilizzare la funzione ***ROL*** per eseguire lo swap su variabile ***DWORD***.

Nell'esempio Il valore ***16#12345678*** viene trasformato nel valore ***16#56781234*** e trasferito nella variabile **VarOut**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarOut	DWORD	Auto	No	0	..	Output variable

Esempio LD

Esempio IL

```
LD  16#12345678
ROL 16
ST  VarOut (* Output variable *)
```

Esempio ST

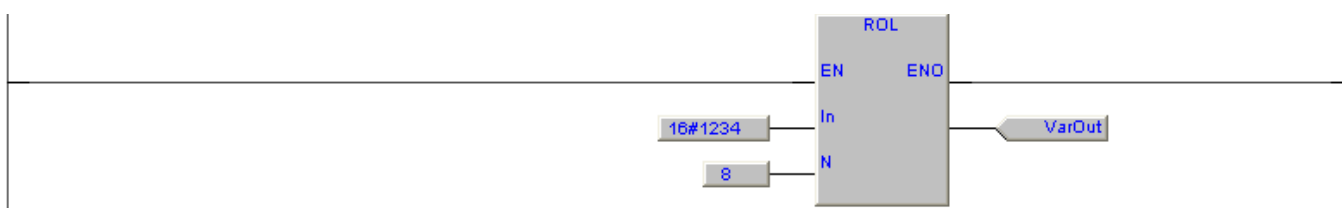
```
VarOut:=ROL(16#12345678, 16); (* Output variable *)
```

10.2 Swap variabile WORD

Ecco come utilizzare la funzione ***ROL*** per eseguire lo swap su variabile ***WORD***.

Nell'esempio Il valore ***16#1234*** viene trasformato nel valore ***16#3412*** e trasferito nella variabile **VarOut**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarOut	WORD	Auto	No	0	..	Output variable

Esempio LD

Esempio IL

```
LD  16#1234
ROL 8
ST  VarOut (* Output variable *)
```

Esempio ST

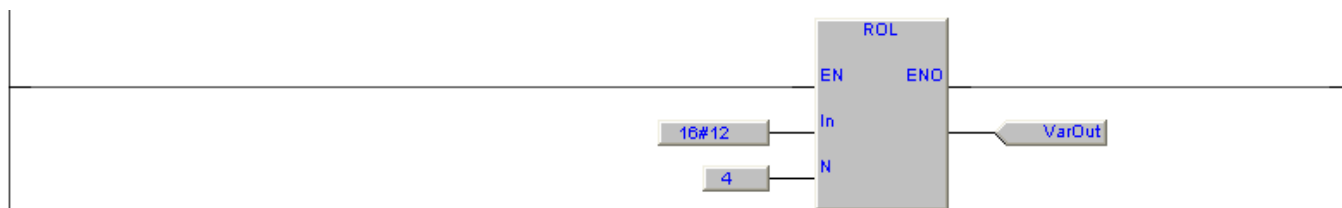
```
VarOut:=ROL(16#1234, 8); (* Output variable *)
```

10.3 Swap variabile BYTE

Ecco come utilizzare la funzione ***ROL*** per eseguire lo swap su variabile ***BYTE***.

Nell'esempio Il valore ***16#12*** viene trasformato nel valore ***16#21*** e trasferito nella variabile **VarOut**.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarOut	BYTE	Auto	No	0	..	Output variable

Esempio LD

Esempio IL

```
LD  16#12
ROL 4
ST  VarOut (* Output variable *)
```

Esempio ST

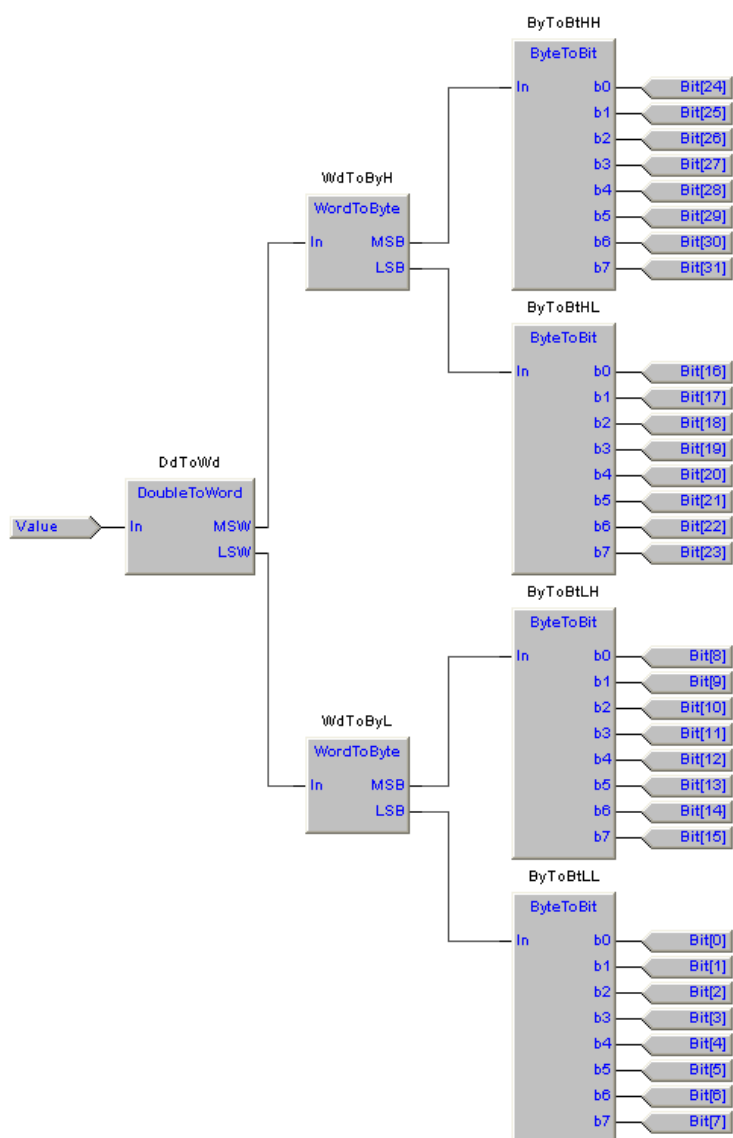
```
VarOut:=ROL(16#12, 4); (* Output variable *)
```

10.4 Espandere DWORD in 32 BOOL

Ecco un come utilizzando i blocchi funzione ***DoubleToWord***, ***WordToByte***, ***ByteToBit*** sia possibile espandere una variabile ***DWORD*** in 32 variabili ***BOOL***.

Definizione variabili

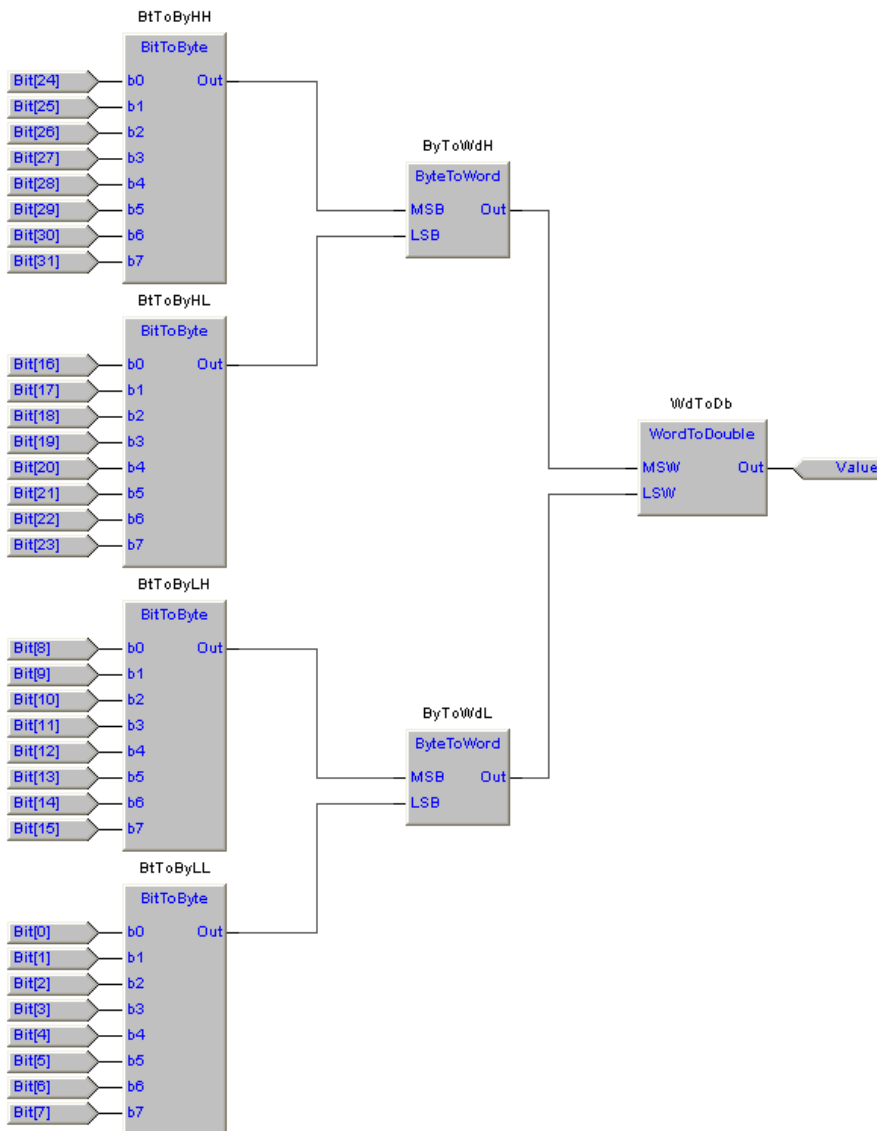
	Name	Type	Address	Array	Init value	Attribute	Description
1	ByToBtHH	ByteToBit	Auto	No	0	..	Byte to bits
2	ByToBtHL	ByteToBit	Auto	No	0	..	Byte to bits
3	ByToBtLH	ByteToBit	Auto	No	0	..	Byte to bits
4	ByToBtLL	ByteToBit	Auto	No	0	..	Byte to bits
5	WdToByH	WordToByte	Auto	No	0	..	Word to bytes
6	WdToByL	WordToByte	Auto	No	0	..	Word to bytes
7	DdToWd	DoubleToWord	Auto	No	0	..	Double to word
8	Bit	BOOL	Auto	[0..31]	FALSE,31(0)	..	Bit status
9	Value	DWORD	Auto	No	0	..	Value to convert

Esempio FBD *(Ptp114a200, FBD_DWExpand)*

10.5 Comprimere 32 BOOL in DWORD

Ecco come utilizzando i blocchi funzione ***BitToByte***, ***ByteToWord***, ***WordToDouble*** sia possibile comprimere 32 variabili ***BOOL*** in una variabile ***DWORD***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
2	BtToByHL	BitToByte	Auto	No	0	..	Bits to byte
3	BtToByLH	BitToByte	Auto	No	0	..	Bits to byte
4	BtToByLL	BitToByte	Auto	No	0	..	Bits to byte
5	ByToWdH	ByteToWord	Auto	No	0	..	Byte to word
6	ByToWdL	ByteToWord	Auto	No	0	..	Byte to word
7	WdToDb	WordToDouble	Auto	No	0	..	Word to double
8	Value	DWORD	Auto	No	0	..	Value to convert
9	Bit	BOOL	Auto	[0..31]	FALSE,31(0)	..	Bit status

Esempio FBD *(Ptp114a200, FBD_DWCompress)*

10.6 Definire caratteri ascii non stampabili

Nella gestione di protocolli di comunicazione e/o per impostare modalità di stampa su stampanti necessita eseguire l'output di caratteri ascii non stampabili, cioè caratteri con codici inferiori al 16#20 o superiori al 16#7F.

Per la definizione dei caratteri ascii stampabili basta includere tra apici singoli i caratteri (Esempio ***'abcd'***).

Per i caratteri non stampabili, occorre anteporre al valore esadecimale del carattere il simbolo $, quindi per il carattere di <STX> 16#02 avremo la definizione ***'$02'***, per <ETX> ***'$03'*** e così di seguito.

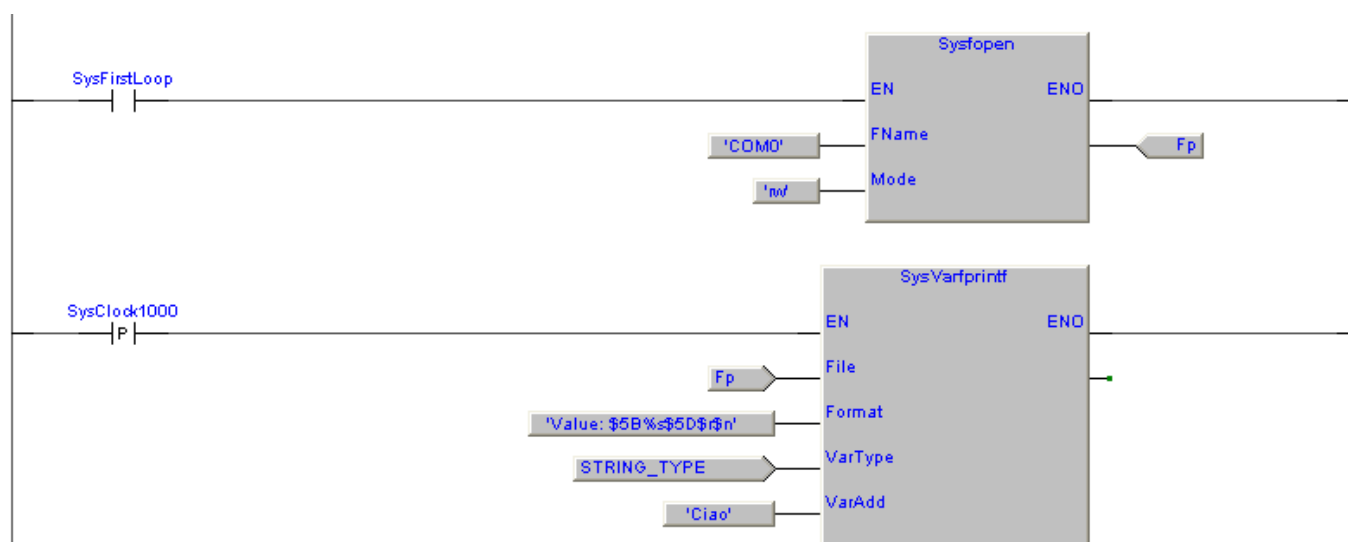
Ricordo che alcuni caratteri di controllo come il line feed, codice 16#0A, sia possibile definirli sia come ***'$0A'*** che come ***'$l'***. Il carriage return, codice 16#0D, è possibile definirlo sia come ***'$0D'*** che come ***'$r'***. Riporto tabella esplicativa.

Sequenza	Significato	Esadecimale	Esempio
$$	Carattere $	16#24	'I paid $$5 for this'
$'	Apostrofo	16#27	'Enter $'Y$' for YES'
$l	Line feed	16#0A	'next $l line'
$r	Carriage return	16#0D	'Hello$r'
$n	New line	16#0D0A	'This is a line$n'
$p	New page	16#0C	'last line $p first line'
$t	Tabulazione	16#09	'name$tsize$tdate'
$hh		16#hh	'ABCD = $41$42$43$44'

Ecco un esempio di utilizzo della funzione ***SysVarfprintf*** per definire oltre ai caratteri stampabili anche i caratteri non stampabili ed inviarli verso lo stream di uscita. In questo esempio viene inviato verso la porta seriale ***COM0*** la stringa ***[Ciao]*** seguita da carriage return e line feed.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	Fp	FILEP	Auto	No	0	..	File pointer

Esempio LD

10.7 Rx/Tx dati su stream

Come si è visto con la funzione ***Sysfopen*** è possibile definire un collegamento tra una risorsa di I/O ed un flusso di dati ***stream*** da cui è possibile gestire la ricezione e/o la trasmissione di dati.

Per la ricezione dei dati in ingresso dallo stream si utilizza la funzione per controllo se dati presenti ***SysGetIChars*** e la funzione per la lettura degli stessi ***Sysfgetc***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	RxString	STRING	Auto	[32]		..	Rx string
2	File	FILEP	Auto	No	0	..	File pointer
3	Ptr	@USINT	Auto	No	0	..	String pointer

Esempio ST

```
(* Rx data from stream. *)

Ptr:=ADR(RxString); (* String pointer *)

WHILE (TO_BOOL(SysGetIChars(File))) DO
    @Ptr:=TO_USINT(Sysfgetc(File)); (* Rx string *)
    Ptr:=Ptr+1; (* String pointer *)

    (* Check if string pointer overflow. *)

    IF (Ptr > ADR(RxString)+31) THEN EXIT; END_IF;
END_WHILE;
```

Per la trasmissione dei dati in uscita dallo ***stream*** si utilizza la funzione che controlla se spazio disponibile ***SysGetOSpace***, e se lo spazio è sufficiente a contenere la stringa, o come nell'esempio, se il buffer di uscita è vuoto è possibile trasferire la stringa sullo stream con la funzione ***SysVarfprintf***.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	TxString	STRING	Auto	[32]		..	Rx string
2	File	FILEP	Auto	No	0	..	File pointer
3	i	UINT	Auto	No	0	..	Auxiliary counter

Esempio ST

```
(* Tx data to stream. *)

IF (TO_UDINT(SysGetOSpace(File)) = SysGetTxBSize(File)) THEN
    i:=TO_UINT(SysVarfprintf(File, '%s', STRING_TYPE, ADR(TxString)));
END_IF;
```

10.8 Conversione tipo dati

La conversione dei dati (Detto anche ***casting***) è una pratica necessaria nella programmazione, naturalmente se il tipo dati di destinazione ha un range inferiore del tipo dati di origine viene effettuato un troncamento del valore. Vediamo caso per caso le varie conversioni:

Tipo BOOL: Le conversioni da qualsiasi tipo verso un **BOOL**, tornano FALSE se il valore del dato da convertire è 0. Tornano TRUE se il valore del dato da convertire è diverso da 0 (anche < 0).

Tipo SINT/USINT: Le conversioni da qualsiasi tipo verso un **USINT**, tornano il valore del byte meno significativo del valore da convertire espresso in esadecimale. Esempio il valore 4660 (16#1234) tornerà 52 (16#34), lo stesso vale per i REAL esempio 300.0 (16#012C) tornerà 44 (16#2C). Per il tipo **SINT** se il valore esadecimale supera 127 il numero sarà negativo.

Tipo INT/UINT: Le conversioni da qualsiasi tipo verso un **UINT**, tornano il valore dei due bytes meno significativi del valore da convertire espresso in esadecimale. Esempio il valore 305419896 (16#12345678) tornerà 22136 (16#5678), lo stesso vale per i REAL esempio 90000.0 (16#15F90) tornerà 24464 (16#5F90). Per il tipo **INT** se il valore esadecimale supera 32767 il numero sarà negativo.

Nella programmazione IEC con LogicLab sono previste apposite funzioni di conversione di tipo, vediamole.

Name	**Input type**	**Output type**	**Function**
DINT_TO_INT	DINT	INT	Converts a long integer (32 bits, signed) into an integer (16 bits, signed)
INT_TO_DINT	INT	DINT	Converts an integer (16 bits, signed) into a long integer (32 bits, signed)
TO_BOOL	Any	BOOL	Converts any data type into a boolean
TO_SINT	Any	SINT	Converts any data type into a short integer (8 bits, signed)
TO_USINT	Any	USINT	Converts any data type into an unsigned short integer (8 bits, unsigned)
TO_INT	Any	INT	Converts any data type into an integer (16 bits, signed)
TO_UINT	Any	UINT	Converts any data type into an unsigned integer (16 bits, unsigned)
TO_DINT	Any	DINT	Converts any data type into a long integer (32 bits, signed)
TO_UDINT	Any	UDINT	Converts any data type into an unsigned long integer (32 bits, unsigned)
TO_REAL	Any	REAL	Converts any data type into a floating point (32 bits, signed)

Esempi

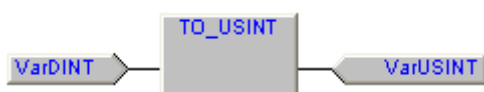
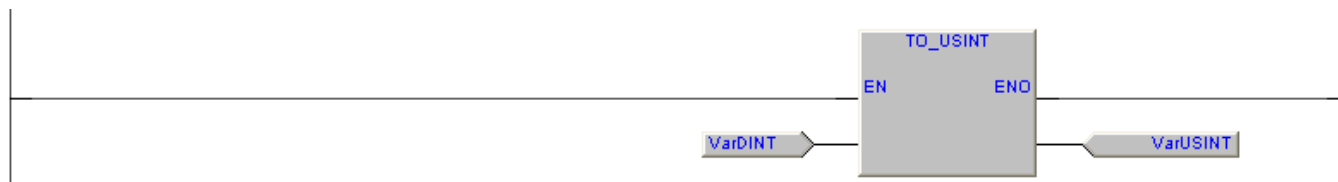
Conversione di una variabile di tipo DINT in una variabile di tipo USINT nei diversi linguaggi di programmazione. Naturalmente se il valore della variabile ***VarDINT*** supera il valore 255 (Limite della variabile ***VarUSINT***), verrà ritornato il resto della divisione per il limite.

Definizione variabili

	Name	Type	Address	Array	Init value	Attribute	Description
1	VarUSINT	USINT	Auto	No	0	..	USINT variable
2	VarDINT	DINT	Auto	No	0	..	DINT variable

Esempio IL

```
LD   VarDINT (* DINT variable *)
TO_USINT
ST   VarUSINT (* USINT variable *)
```

Esempio FBD

Esempio LD

Esempio ST

```
VarUSINT:=TO_USINT(VarDINT); (* USINT variable *)
```

10.9 User Informations and Settings

E' previsto un interfacciamento tra il programma utente PLC ed il sistema operativo, questo permette di visualizzare ed impostare da sistema operativo variabili il cui valore è disponibile all'interno del programma utente.

SysUInfoA, ***SysUInfoB***, ***SysUInfoC***, ***SysUInfoD***, 4 variabili stringa da 16 caratteri, che possono essere valorizzate da programma utente e visualizzabili da sistema operativo.

SysUSetA, ***SysUSetB***, ***SysUSetC***, ***SysUSetD***, 4 variabili stringa da 16 caratteri, che possono essere valorizzate da sistema operativo e utilizzate da programma utente.

E' prevista una pagina web (Menù ***User***) in cui sono visualizzate le variabili ***SysUInfo(x)*** ed è possibile impostare le variabili ***SysUSet(x)***.

ELSIST - SlimLine

Home | General Setup | Time Setup | User

User informations and settings

(x)	SysUInfo(x)	SysUSet(x)
A	12	12
B	34	34
C	56	56
D	78	78

Read Write

ELSIST - SlimLine
For more information visit our website: www.elsist.it

Nell'esempio riporto un semplice programma che acquisisce i valori delle variabili ***SysUSet(x)*** e li trasferisce nella rispettiva variabile di visualizzazione ***SysUInfo(x)***. Nella pagina web il valore impostato nella variabile ***SysUSet(x)*** su accettazione con il tasto ***Write*** verrà ritornato nella variabile ***SysUInfo(x)*** come visibile dallo screenshot riportato sopra.

Definizione variabili

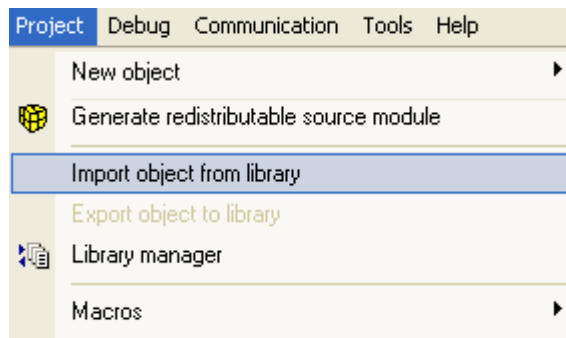
	Name	Type	Address	Array	Init value	Attribute	Description
1	Values	UDINT	Auto	[0..3]	4(0)	..	User variables value
2	i	INT	Auto	No	0	..	Auxiliary variable

Esempio ST

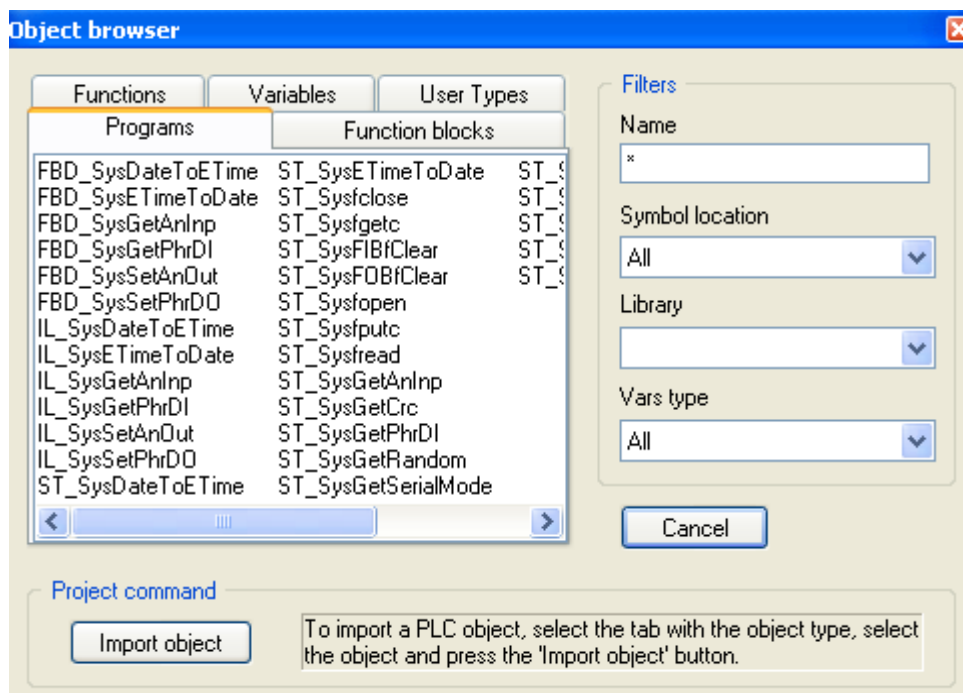
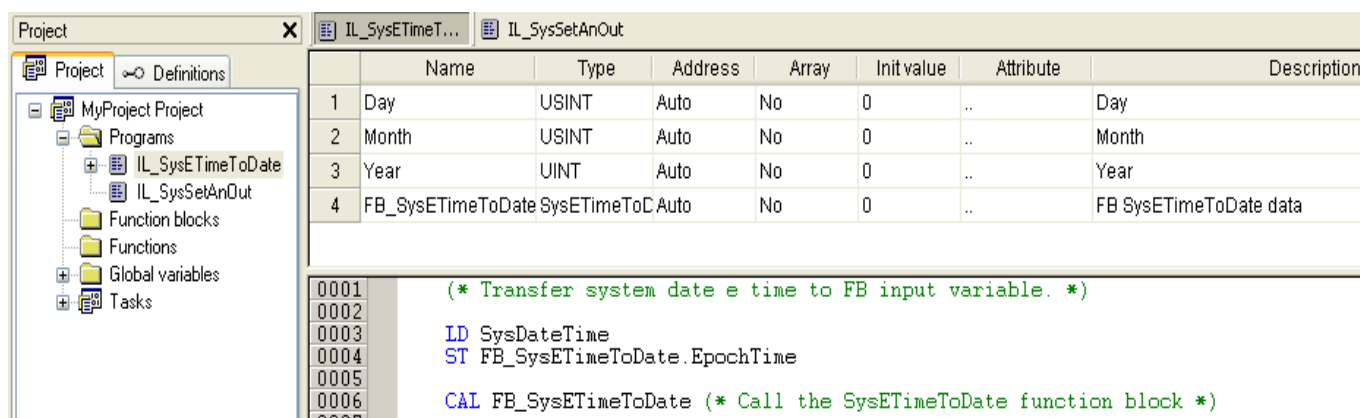
```
(* Read user settings and write user infos. *)

IF (SysUVSet) THEN
    i:=SysVarsscanf(ADR(SysUSetA), '%d', UDINT_TYPE, ADR(Values[0]));
    i:=SysVarsnprintf(ADR(SysUInfoA), 16, '%d', UDINT_TYPE, ADR(Values[0]));

    i:=SysVarsscanf(ADR(SysUSetB), '%d', UDINT_TYPE, ADR(Values[1]));
    i:=SysVarsnprintf(ADR(SysUInfoB), 16, '%d', UDINT_TYPE, ADR(Values[1]));

    i:=SysVarsscanf(ADR(SysUSetC), '%d', UDINT_TYPE, ADR(Values[2]));
    i:=SysVarsnprintf(ADR(SysUInfoC), 16, '%d', UDINT_TYPE, ADR(Values[2]));

    i:=SysVarsscanf(ADR(SysUSetD), '%d', UDINT_TYPE, ADR(Values[3]));
    i:=SysVarsnprintf(ADR(SysUInfoD), 16, '%d', UDINT_TYPE, ADR(Values[3]));
END_IF;
```

11 Esempi di programmazione

11.1 Biblioteca esempi

Per permettere all'utente di disporre di esempi da utilizzare per lo sviluppo dei proprii programmi quasi tutti gli esempi riportati sul manuale sono forniti in programmi dimostrativi. I programmi dimostrativi sono codificati con il suffisso PTP, accanto ad ogni esempio . Se si desidera includere nel proprio progetto un file di esempio del manuale occorre dal menù ***Project*** scegliere la voce ***Import object from library***. Si aprirà un dialog box che permette di selezionare la libreria da cui estrarre il programma da importare.

Scegliere il file di libreria desiderato (Esempio ***Ptp116*000.pll***) si aprirà la finestra con l'elenco degli oggetti presenti all'interno da cui sarà possibile selezionare gli oggetti desiderati.

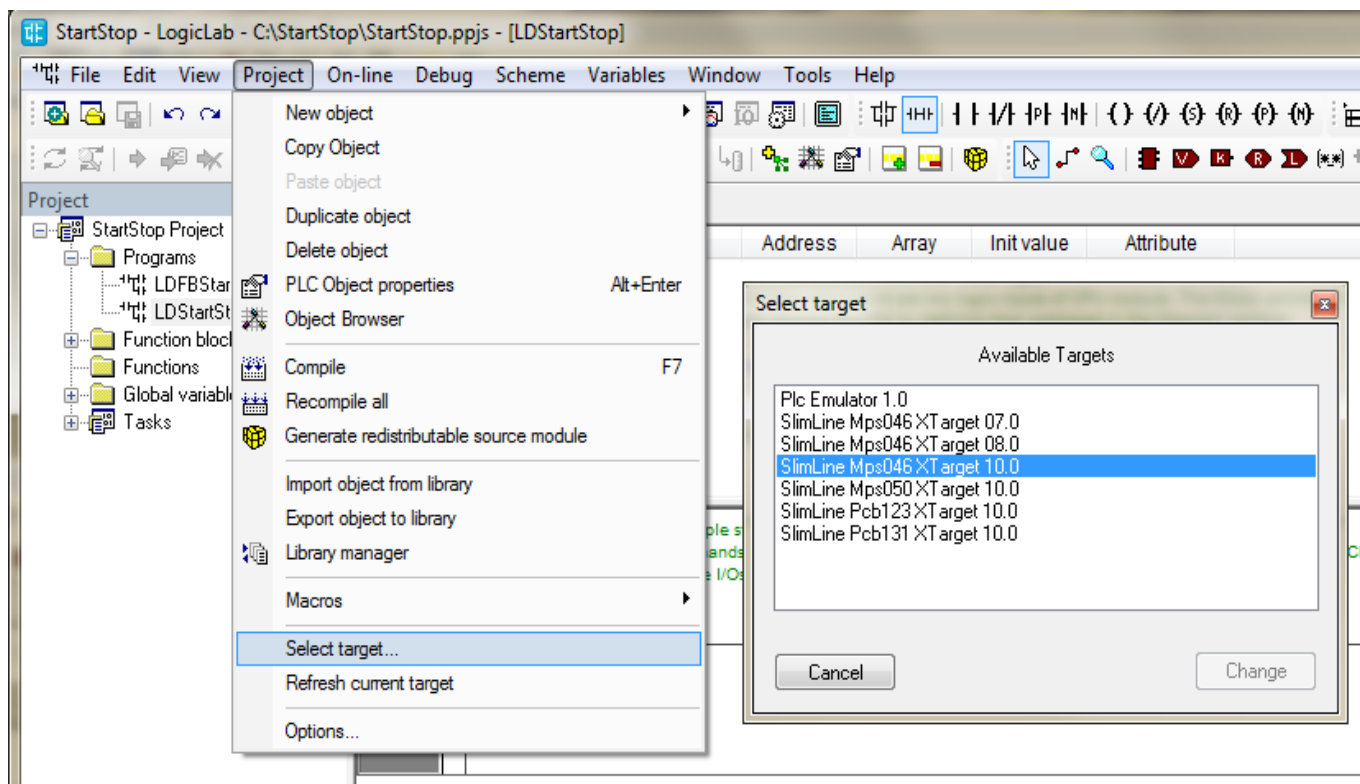
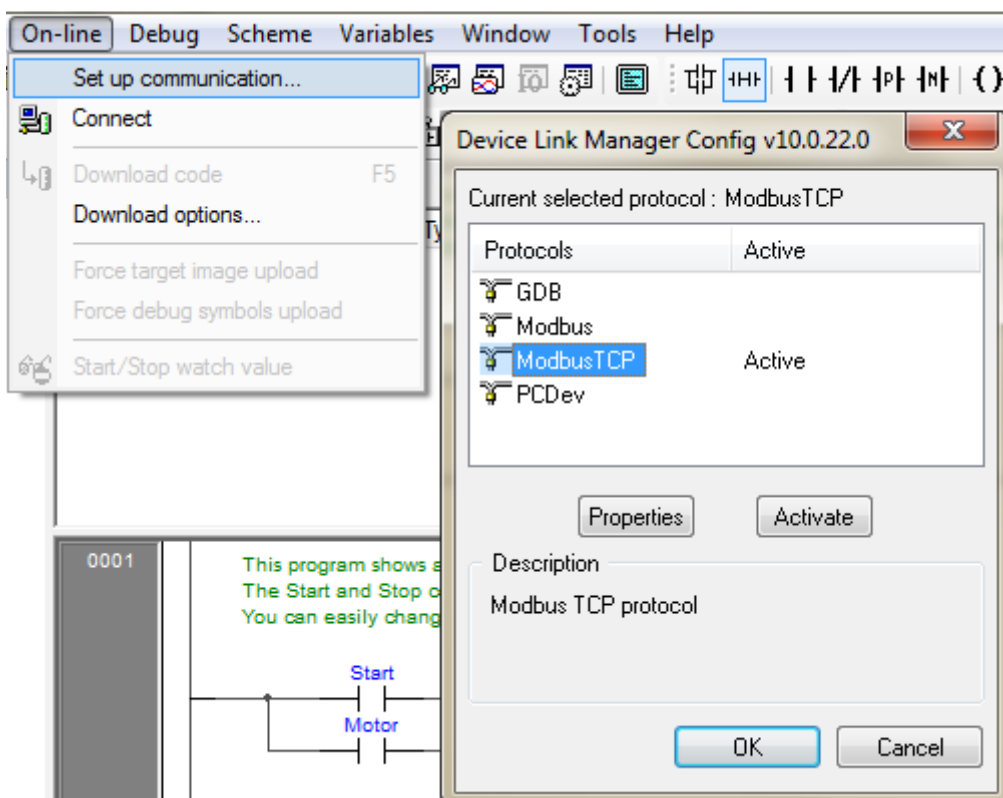
Evidenziando gli oggetti e agendo sul tasto ***Import Object***, gli oggetti selezionati saranno inclusi nel proprio progetto.

Oltre ai programmi è possibile importare dalla libreria anche le variabili. In questo modo si potranno importare tutte le definizioni degli I/O logici come indicato nella tabella di definizione.

Una volta inclusi nel progetto gli esempi, sarà possibile utilizzarli direttamente, oppure con con semplici operazioni di cut and paste incollare parti di codice sorgente dal progetto di esempio.

11.2 Definizioni I/O logici negli esempi

Tutti gli esempi riportati in questo manuale sono stati realizzati utilizzando un sistema configurato con un modulo CPU SlimLine tipo MPS046A100 abbinato ad un modulo Mixed I/O PCB122*100 (Impostato con address 0). Tutti gli I/O del modulo sono stati abbinati a variabili mnemoniche. Gli ingressi sono denominati ***Di0xM00*** e le uscite ***Do0xM00*** come evidente nella tabella di definizione.

	Name	Type	Address	Group	Array	Init value	Attribute	Description
1	Di00M00	BOOL	%IX0.0		No	FALSE	..	Input 00, Module address 0
2	Di01M00	BOOL	%IX0.1		No	FALSE	..	Input 01, Module address 0
3	Di02M00	BOOL	%IX0.2		No	FALSE	..	Input 02, Module address 0
4	Di03M00	BOOL	%IX0.3		No	FALSE	..	Input 03, Module address 0
5	Di04M00	BOOL	%IX0.4		No	FALSE	..	Input 04, Module address 0
6	Di05M00	BOOL	%IX0.5		No	FALSE	..	Input 05, Module address 0
7	Di06M00	BOOL	%IX0.6		No	FALSE	..	Input 06, Module address 0
8	Di07M00	BOOL	%IX0.7		No	FALSE	..	Input 07, Module address 0
9	Di08M00	BOOL	%IX0.8		No	FALSE	..	Input 08, Module address 0
10	Di09M00	BOOL	%IX0.9		No	FALSE	..	Input 09, Module address 0
11	Di10M00	BOOL	%IX0.10		No	FALSE	..	Input 10, Module address 0
12	Di11M00	BOOL	%IX0.11		No	FALSE	..	Input 11, Module address 0
13	Do00M00	BOOL	%QX0.0		No	FALSE	..	Output 00, Module address 0
14	Do01M00	BOOL	%QX0.1		No	FALSE	..	Output 01, Module address 0
15	Do02M00	BOOL	%QX0.2		No	FALSE	..	Output 02, Module address 0
16	Do03M00	BOOL	%QX0.3		No	FALSE	..	Output 03, Module address 0
17	Do04M00	BOOL	%QX0.4		No	FALSE	..	Output 04, Module address 0
18	Do05M00	BOOL	%QX0.5		No	FALSE	..	Output 05, Module address 0
19	Do06M00	BOOL	%QX0.6		No	FALSE	..	Output 06, Module address 0
20	Do07M00	BOOL	%QX0.7		No	FALSE	..	Output 07, Module address 0

11.3 Esempi forniti con LogicLab

Nella distribuzione del programma LogicLab sono inseriti alcuni programmi di esempio, i programmi sono forniti in codice sorgente e possono essere trasferiti sul sistema target e provati. Nell finestra principale di LogicLab sotto la voce ***Example projects*** vi è il link ai vari progetti. Per poter utilizzare un esempio sul proprio sistema target occorre definire il sistema con il quale si stà operando dal menù **Project** → **Select target**.

E definire il modo di comunicazione utilizzato dal menù **On-line** → **Set up communication**.

11.3.1 Elenco programmi di esempio

I programmi forniti con LogicLab sono:

StartStop	Logica di marcia/arresto
SMSbyWeb	Invio messaggi SMS da pagina web
MdbAsciiMaster	FB gestione protocollo Modbus Ascii (Modo Master)
MultipleSTE	Connessione in SNMP con alcuni dispositivi STE
EasyProtocol	Sviluppo di un semplice protocollo di comunicazione
PowerOneCm	Comunicazione con inverter Aurora della Power One
TagReader	Controllo accessi con TAG I-button
GSMDoorOpen	Apricancello su chiamata telefonica
TCPAsciiProtocol	Semplice comunicazione ascii su connessione TCP/IP
CSVFileTimeSwitch	Programmatore orario con lettura tempi da file CSV
SineWave	Generatore onda sinusoidale

12 Appendici

12.1 Tabella istruzioni IL

Istruzione	Operandi	Descrizione
LD	Tutti	Carica il valore operando nell'accumulatore
LDN	Tutti	Carica il valore negato operando nell'accumulatore
ST	Tutti	Trasferisce il valore dell'accumulatore nell'operando
STN	Tutti	Trasferisce il valore negato dell'accumulatore nell'operando
S	BOOL	Setta l'operando (Accetta solo BOOL) se l'accumulatore è TRUE
R	BOOL	Resetta l'operando (Accetta solo BOOL) se l'accumulatore è TRUE
AND	Tutti meno REAL	AND a bit tra accumulatore e valore operando, risultato in accumulatore
ANDN	Tutti meno REAL	AND a bit tra accumulatore e valore negato operando, risultato in accumulatore
OR	Tutti meno REAL	OR a bit tra accumulatore e valore operando, risultato in accumulatore
ORN	Tutti meno REAL	OR a bit tra accumulatore e valore negato operando, risultato in accumulatore
XOR	Tutti meno REAL	XOR a bit tra accumulatore e valore operando, risultato in accumulatore
XORN	Tutti meno REAL	XOR a bit tra accumulatore e valore negato operando, risultato in accumulatore
NOT		Esegue l'inversione a bit del valore in accumulatore
ADD	Tutti meno BOOL	Somma tra accumulatore e valore operando, risultato in accumulatore
SUB	Tutti meno BOOL	Sottrazione tra accumulatore e valore operando, risultato in accumulatore
MUL	Tutti meno BOOL	Moltiplicazione tra accumulatore e valore operando, risultato in accumulatore
DIV	Tutti meno BOOL	Divisione tra accumulatore e valore operando, risultato in accumulatore
MOD	Tutti meno BOOL	Ritorna il modulo della divisione nell'accumulatore
GT	Tutti meno BOOL	Controlla se accumulatore > operando, risultato (BOOL) in accumulatore
GE	Tutti meno BOOL	Controlla se accumulatore >= operando, risultato (BOOL) in accumulatore
EQ	Tutti meno BOOL	Controlla se accumulatore = operando, risultato (BOOL) in accumulatore
NE	Tutti meno BOOL	Controlla se accumulatore <> operando, risultato (BOOL) in accumulatore
LE	Tutti meno BOOL	Controlla se accumulatore <= operando, risultato (BOOL) in accumulatore
LT	Tutti meno BOOL	Controlla se accumulatore < operando, risultato (BOOL) in accumulatore
JMP	Etichetta	Salta incondizionatamente su etichetta
JMPC	Etichetta	Salta su etichetta se accumulatore diverso da zero
JMPCN	Etichetta	Salta su etichetta se accumulatore uguale a zero
CAL	FB	Esegue incondizionatamente il blocco funzione
CALC	FB	Esegue blocco funzione se accumulatore diverso da zero
CALCN	FB	Esegue blocco funzione se accumulatore uguale a zero
RET		Ritorna incondizionatamente al programma che ha eseguito CALL
RETC		Ritorna al programma che ha eseguito CALL se accumulatore diverso da zero

12.2 Operatori linguaggio ST

Nella tabella seguente sono riportati gli operatori utilizzabili nel linguaggio ST. Gli operatori sono riportati in tabella in base alla loro priorità, dall'alto verso il basso, quindi le parentesi hanno priorità maggiore su tutti gli altri operatori.

Operatore	**Simbolo**	**Esempio**
Parenthesization	(Espressione)	
Function evaluation	Funzione(Argomenti)	`LN(A), MAX(X,Y), etc.`
Negation	-	
Complement	NOT	
Exponentiation	**	
Multiply	*	
Divide	/	
Modulo	MOD	
Add	+	
Subtract	-	
Comparison	< , > , <= , >=	
Equality	=	
Inequality	<>	
Boolean AND	&	
Boolean AND	AND	
Boolean Exclusive OR	XOR	
Boolean OR	OR	

12.3 Statements linguaggio ST

Nella tabella seguente sono riportati gli operatori utilizzabili nel linguaggio ST. Gli operatori sono riportati in tabella in base alla loro priorità, dall'alto verso il basso, quindi le parentesi hanno priorità maggiore su tutti gli altri operatori.

Statement	**Esempio**
Assignment	`A:=B; CV:=CV+1; C:=SIN(X);`
FB Invocation and output usage	`CMD_TMR(IN:=%IX5, PT:=T#300ms);` `A:=CMD_TMR.Q;`
RETURN	`RETURN;`

IF	`D:=B*B-4*A*C;`
	
	`IF D < 0.0 THEN NROOTS:=0;`
	`ELSIF D=0.0 THEN`
	`    NROOTS:=1;`
	`    X1:=-B/(2.0*A);`
	`ELSE`
	`    NROOTS:=2;`
	`    X1:=(-B+SQRT(D))/(2.0*A);`
	`    X2:=(-B-SQRT(D))/(2.0*A);`
	`END_IF;`

CASE	`TW:=BCD_TO_INT(THUMBWHEEL);`
	`TW_ERROR:=0;`
	
	`CASE TW OF`
	`    1,5: DISPLAY:=OVEN_TEMP;`
	`    2: DISPLAY:=MOTOR_SPEED;`
	`    3: DISPLAY:=GROSS-TARE;`
	`    4,6..10: DISPLAY:=STATUS(TW-4);`
	`ELSE`
	`    DISPLAY:=0;`
	`    TW_ERROR:=1;`
	`END_CASE;`
	
	`QW100:=INT_TO_BCD(DISPLAY);`

FOR	`J:=101;`
	
	`FOR I:=1 TO 100 BY 2 DO`
	`    IF WORDS[I]='KEY' THEN`
	`        J:=I;`
	`        EXIT;`
	`    END_IF;`
	`END_FOR;`

WHILE	`J:=1;`
	
	`WHILE J <= 100 & WORDS[J]<>'KEY' DO`
	`    J:=J+2;`
	`END_WHILE;`

REPEAT	`J:=-1;`
	
	`REPEAT`
	`    J:=J+2;`
	`UNTIL J=101 OR WORDS[J]='KEY'`
	`END_REPEAT ;`

EXIT	`EXIT;`
Empty Statement	`;`

12.4 ID oggetti

Tutti gli oggetti (Funzioni e/o blocchi funzione) hanno un identificativo univoco che li contraddistingue. L'ID è ritornato anche nell'eventuale codice di errore (Riferimento alla funzione ***SysGetLastError*** ed alla variabile ***SysLastError***). Il codice di errore si compone dell'ID oggetto seguito da 3 cifre che indicano il codice di errore.

Di seguito tabella degli ID oggetti (Rif. MDP070xx00), il simbolo (●) indica che l'oggetto è obsoleto. Sugli oggetti obsoleti è riportato tra parentesi l'ultima versione di libreria e di manuale in cui l'oggetto era presente.

ID	Oggetto
9947 (000 ÷ 999)	SysMAlloc, memory allocation
9948 (000 ÷ 999)	SysSetTaskLpTime, set task loop time
9949 (000 ÷ 999)	Funzione embedded
9950 (000 ÷ 999)	SysSpyData, system spy data
9951 (000 ÷ 999)	SysSetPWMOut, set PWM output
9952 (000 ÷ 999)	SysDirListing, directory listing
9953 (000 ÷ 999)	SysI2CWrRd, writes/reads on I2C extension bus
9954 (000 ÷ 999)	SysCANTxMsg, transmit a CAN message
9955 (000 ÷ 999)	SysCANRxMsg, receives a CAN message
9956 (000 ÷ 999)	SysIsCANRxTxAv, checks if CAN Rx or Tx is available
9957 (000 ÷ 999)	SysCANSetMode, set the CAN controller mode
9958 (000 ÷ 999)	Sysfseek, file seek
9959 (000 ÷ 999)	Sysfilelength, file lengh
9960 (000 ÷ 999)	Sysrename, file rename
9961 (000 ÷ 999)	Sysremove, file remove
9962 (000 ÷ 999)	SysFOBfFlush, file output buffer flush
9963 (000 ÷ 999)	SysFOBfClear, file output buffer clear
9964 (000 ÷ 999)	SysFIBfClear, file input buffer clear
9965 (000 ÷ 999)	SysGetTxBSize, get file Tx output buffer size
9966 (000 ÷ 999)	SysGetRxBSize, get file Rx input buffer size
9967 (000 ÷ 999)	SysGetOSpace, get output available space on file
9968 (000 ÷ 999)	SysGetIChars, get input available characters from file
9969 (000 ÷ 999)	Sysfwrite, write data to file
9970 (000 ÷ 999)	Sysfread, read data from file
9971 (000 ÷ 999)	Sysfputc, put character to file
9972 (000 ÷ 999)	Sysfgetc, get character from file
9973 (000 ÷ 999)	Sysfclose, file close
9974 (000 ÷ 999)	SysIPReach, IP address is reachable
9975 (000 ÷ 999)	SysUDPSktRcv, UDP socket receive
9976 (000 ÷ 999)	SysUDPSktSend, UDP socket send
9977 (000 ÷ 999)	SysSktListen, socket listen
9978 (000 ÷ 999)	SysGetCrc, get CRC value
9979 (000 ÷ 999)	SysDMXMng, DMX management
9980 (000 ÷ 999)	SysGetEncoder, get encoder input
9981 (000 ÷ 999)	SysGetCounter, get counter
9982 (000 ÷ 999)	SysSetAnOut, set analog output
9983 (000 ÷ 999)	SysGetAnInp, get analog input
9984 (000 ÷ 999)	SysSetPhrDO, set peripheral digital output
9985 (000 ÷ 999)	SysGetPhrDI, get peripheral digital input
9986 (000 ÷ 999)	SysETimeToDate, epoch time to date conversion
9987 (000 ÷ 999)	SysDateToETime, date to epoch time conversion
9988 (000 ÷ 999)	SysPhrVWr, write variable to peripheral module

9989 (000 ÷ 999) SysPhrVRd, read variable from peripheral module
9990 (000 ÷ 999) SysPhrInfos, get infos from peripheral modules
9991 (000 ÷ 999) SysPCodeAccept, accepts the protection code
9992 (000 ÷ 999) SysSetSerialDTR, set DTR signal status
9993 (000 ÷ 999) SysGetSerialCTS, get serial CTS signal status
9994 (000 ÷ 999) SysSetSerialMode, set serial mode
9995 (000 ÷ 999) SysGetSerialMode, get serial mode
9996 (000 ÷ 999) Sysfopen, file open
9997 (000 ÷ 999) SysVarsnprintf, variable print to string
9998 (000 ÷ 999) SysVarfprintf, variable print to file
9999 (000 ÷ 999) SysVarsscanf, extracts values from string

- 10000 (000 ÷ 999) MDBRTUMASTER, modbus Rtu master (ePLCUtyLib_A400, MNL151B040)

10001 (000 ÷ 999) CPUModuleIO, CPU module I/O management
10002 (000 ÷ 999) ModemCore_v3, modem core management
10003 (000 ÷ 999) ModemSMSReceive, receive a SMS message
10004 (000 ÷ 999) ModemSMSRxCmd_v1, receive a SMS command
10005 (000 ÷ 999) FblModemSMSSend_v2, send a SMS message
10006 (000 ÷ 999) SetSMode, Set serial mode
10007 (000 ÷ 999) ModbusMaster, modbus RTU master
10008 (000 ÷ 999) OwireMng, One-Wire management
10009 (000 ÷ 999) OWRdIdentifier, One-Wire read ROM identifier
10010 (000 ÷ 999) OWRdTemperature, One-Wire read temperature
10011 (000 ÷ 999) IODataExchange, exchange data by using logic I/O
10012 (000 ÷ 999) PIDMng, PID management
10013 (000 ÷ 999) STESnmpAcq, STE termometer acquisition over SNMP
10014 (000 ÷ 999) UDPDataTxfer, UDP data transfer
10015 (000 ÷ 999) OWRdHumidity, One-Wire read humidity
10016 (000 ÷ 999) IEC62056_21Rd, IEC62056-21 protocol read
10017 (000 ÷ 999) NMEASInterface, NMEA system interface
10018 (000 ÷ 999) GLLSentence, Geographic Position sentence

- 10019 (000 ÷ 999) ModbusRTUSlave, modbus Rtu slave (ePLCUtyLib_B000, MNL151C020)

10020 (000 ÷ 999) MWVSentence, Wind Speed and Angle sentence
10030 (000 ÷ 999) AuroraDSPMeasure, Aurora measure request to DSP
10031 (000 ÷ 999) AuroraCEnergy, Aurora cumulated energy reading
10032 (000 ÷ 999) sHWgSProtocol, HW group serial protocol

- 10033 (000 ÷ 999) ModbusAsciiSlave, modbus Ascii slave (ePLCUtyLib_B000, MNL151C020)

10034 (000 ÷ 999) SysLogReport, send a report to Syslog server
10035 (000 ÷ 999) StringToLogFile, store string to a log file
10036 (000 ÷ 999) FileMemoryDump, dump memory on file
10037 (000 ÷ 999) ModemPhoneCall_v1, executes a phone call
10038 (000 ÷ 999) ModbusSlave, modbus slave
10039 (000 ÷ 999) HIDClkDtaReader, HID RFID clock/data reader
10040 (000 ÷ 999) MMasterDataTxfer, multimaster data transfer
10041 (000 ÷ 999) DataTxferClient, Data transfer client
10042 (000 ÷ 999) ModemHTTPGet, executes a HTTP Get request
10043 (000 ÷ 999) SpyDataFile, spy data and stores them on a file
10044 (000 ÷ 999) BroadcastDataSend, broadcast data send
10045 (000 ÷ 999) StrainGaugeAcq, strain gauge acquisition
10046 (000 ÷ 999) HMIBuiltInMessages, HMI built in messages

10047 (000 ÷ 999) HMIBuiltInNetlog, Netlog HMI management
10048 (000 ÷ 999) DataStreamExch, exchanges data between two I/O streams

12.5 Tabella codici Ascii

12.5.1 Tabella codici ASCII standard

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

12.5.2 Tabella codici ASCII estesi

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	□